all rights reversed

HACKER

Μάρτης 2014 # Τιμή τεύχους 4€

τεύχος 030 - Ανάμικτες Καραμέλες Edition

είναι ο κόσμος μου ΑΛΗΘΙΝΟΣ;*

Επιθέσεις με το Sqlmap
Rsync, ο κρυφός σας σύμμαχος
Ανάπτυξη web application με βάση δεδομένων

[*] ...ή πώς κάποια malware κρύβονται αποτελεσματικά από τους αναλυτές!

all rights reversed

HACKER

tefhos 30o

Περιεχόμενα

και οδηγίες χρήσης

56

Web development σε περιβάλλον LAMP [μέρος 8]

Όσον αφορά στην ανάπτυξη web εφαρμογών σε περιβάλλον LAMP stack, οι ορίζοντές μας έχουν διευρυνθεί αρκετά. Στο τελευταίο άρθρο της σειράς μας εξετάζουμε ένα κοινότοπο χαρακτηριστικό των web εφαρμογών, το οποίο δεν είναι άλλο από την ταυτοποίηση των χρηστών.

70

SQL injections: Εξοικειωθείτε με τον κίνδυνο!

Δεν σας προτείνουμε ν' αποδεχτείτε μοιρολατρικά την ιδέα του κινδύνου, ούτε ν' αντιμετωπίσετε το ενδεχόμενο μιας επίθεσης με επιπολαιότητα. Θέλουμε όμως να γνωρίσετε τις μεθόδους και τα όπλα των επιτιθέμενων, ώστε να είσαστε άψογα προετοιμασμένοι *πριν* ακόμα δεχτείτε χτύπημα...

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Γιώργος Αραβίδης, Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Ανδρέας Βενιέρης, Γιώργος Γιάννου, Μανώλης Κιαγιάς, Ιωάννης Κονιάρης, Πέτρος Κυλαδίτης, Γιώργος Μελέκος, Νίκος Μουρατίδης, Νικόλαος Σηφάκης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στη Μαιώτιδος 3, 55133, Καλαμαριά. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/subscriptions και οι ίδιες οι παραγγελίες γίνονται από το http://deltahacker.gr/pdf12order. Όλες οι απόψεις που εκφράζονται στα άρθρα δεν εκφράζουν απαραίτητα και τη γνώμη του περιοδικού.

(*) **deltaCast s02e02 | notMe-CryptoBSD**
http://deltahacker.gr/deltacast-s02e02


delta HACKER
Στη 3η χρονιά και συνεχίζουμε ακάθεκτοι!
Μηνιαίο περιοδικό deltaHacker, http://deltahacker.gr
Ethical Hacking / Δίκτυα / Ασφάλεια / Προγραμματισμός / Ηλεκτρονικά
twitter.com/deltaHacker
facebook.com/deltaHacker

Του κεφαλιού μας

Αγαπητές κι αγαπητοί,

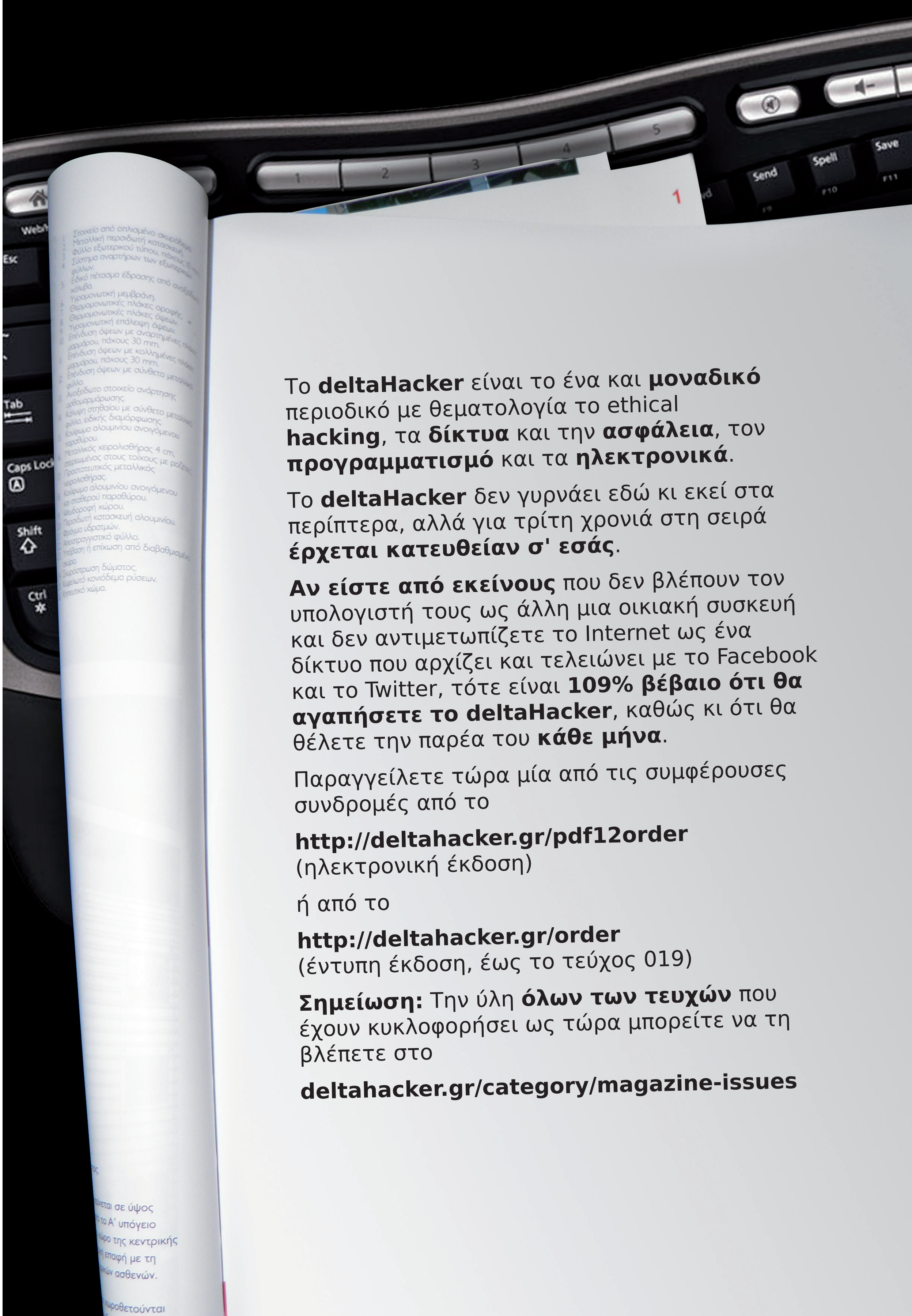
- Ξεκινάω το editorial του τεύχους 030 όχι μ' ένα σκέτο "αγαπητοί μου".
- Θέλω με αυτόν τον τρόπο να διαφοροποιηθώ από τον Θείο Ακάκιο, ο οποίος ξεκινά πάντα τις επιστολές του με ένα "αγαπητοί μου" — σκέτο.
- Σας το λέω επίσης από τώρα, ότι αντί να τελειώσω το editorial με το "σας ασπάζομαι", το οποίο έχει καθιερώσει ο Θείος, εγώ θα το κλείσω μ' ένα "καλά να περνάτε".
- Να, μόλις πήγα τώρα στο κάτω μέρος του εγγράφου (του Γουόρντ) και το πληκτρολόγησα, ώστε πιο μετά να μην το ξεχάσω.
- Βέβαια εσείς δεν μπορείτε να είστε σίγουροι για το αν πράγματι πήγα στο τέλος του εγγράφου (του Γουόρντ) για να το πληκτρολογήσω, φαντάζομαι όμως ότι, τουλάχιστον οι παλιοί μας συνδρομητές, μας ξέρετε αρκετά ώστε να μας πιστεύετε.
- Ένα άλλο σημείο στο οποίο θέλω να διαφοροποιηθώ από τον Θείο Ακάκιο, είναι ότι *και* στο περιοδικό θα ξεκινώ κάθε παράγραφο του editorial με bullet. Όπως, δηλαδή, κάνω και στο site του περιοδικού.
- Μάλιστα δεν θα ενημερώσω σχετικά τον art director της Parabing Creations, αφού πριν ξεκινήσει το στήσιμο του —οποιουδήποτε— άρθρου θα το διαβάσει πρώτα.
- Πάντα έτσι κάνει, σε αντίθεση με κάθε άλλον art director με τον οποίο έχω συνεργαστεί.
- Πολύ ενδιαφέρον μου φαίνεται εμένα αυτό — ίσως όμως να φταίει και το γεγονός ότι δεν έχω συνεργαστεί με πολλούς art directors.
- Τέλος πάντων.
- Οφείλω να ομολογήσω ότι το να γράφεις τις σκέψεις σου σε σύντομες, φαινομενικά ψιλο-άσχετες μεταξύ τους παραγράφους, διευκολύνει σημαντικά.
- Αν μη τι άλλο, αποτελεί ένα εξαιρετικά ευχάριστο διάλειμμα από τα σεντόνια που συχνά-πυκνά γράφουμε για το deltaHacker.
- Ευτυχώς δηλαδή που έχουμε και το deltaCast, το video podcast του περιοδικού, γιατί διαφορετικά μάλλον θα είχατε απηυδήσει να διαβάζετε τα κατεβατά μας.
- Βέβαια στο deltaCast δεν έχουμε κατεβατά κειμένων —είπαμε, video podcast είναι αυτό—, τα ίδια τα επεισόδια όμως ταινίες ολόκληρες θυμίζουν.

- Τουλάχιστον όσον αφορά στη διάρκεια.
- Πού ακούστηκε, αγαπητές κι αγαπητοί, τεχνικό video podcast με επεισόδια δύο και πλέον ωρών;
- Να σας πω εγώ πού ακούστηκε: Πουθενά, από κανέναν.
- Εκτός από τα ίδια τα επεισόδια του deltaCast.
- Είναι βλέπετε που, για άλλη μια φορά, κάνουμε του κεφαλιού μας.
- Άλλοι στη θέση μας, από ένα μόνο επεισόδιο θα έβγαζαν τρία διαφορετικά.
- Εμείς, όμως; Όοοοοχι! Αντί για τρία διαφορετικά, σαραντάλεπτα επεισόδια, βγάζουμε ένα μονοκόμματο.
- Κι όποιος αντέξει.
- Αν κρίνουμε πάντως από το feedback, οι περισσότεροι φαίνεται να αντέχουν.
- Τέλος πάντων.
- Το ηθικό δίδαγμα εδώ είναι ότι, γενικά, κάνουμε του κεφαλιού μας.
- Ακόμη και η κυκλοφορία του ίδιου του deltaHacker, αποτέλεσμα της ίδιας κοσμοαντίληψης είναι. Αυτής: Κάνω αυτό που νομίζω πιο σωστό, χωρίς κατ' ανάγκη ν' ακολουθώ την πεπατημένη — και λίγο πολύ αγνοώντας τις συμβουλές των "συνετών".
- Τι έλεγαν οι συνετοί, στις αρχές του καλοκαιριού του '11;
- "Πού πάτε να μπλέξετε τώρα με νέο περιοδικό, χωρίς κεφάλαιο και μέσα σ' αυτή την κρίση; Καθίστε στ' αυγά σας, γιατί θα χάσετε ό,τι έχετε και δεν έχετε."
- Δεν μπορώ να πω ότι *δεν* δώσαμε σημασία σε παροτρύνσεις σαν αυτή.
- Απλά, συνολικά δεν πρέπει να τους δώσαμε σημασία για πάνω από 40 λεπτά — και πολύ λέω, δηλαδή.
- Κάναμε τελικά του κεφαλιού μας και, περίπου τρία χρόνια μετά, να 'μαστε εδώ, να τα λέμε όμορφα κι ωραία και να κυκλοφορούμε ένα τεχνικό περιοδικό που όμοιό του δεν υπάρχει.
- Και προσέξτε, όταν λέω "όμοιό του δεν υπάρχει", το εννοώ με την καλή έννοια :D
- Ανάλογες παρατηρήσεις μπορώ να κάνω και για το deltaCast. Βέβαια ένα video podcast δεν είναι το ίδιο πράγμα με ένα περιοδικό, αλλά εμείς έχουμε μια ελαφρώς διαφορετική οπτική.
- Πρώτα, ρωτάμε εαυτούς: Τι είναι αυτό που κάνουμε;
- Απαντάμε: Παράγουμε περιερχόμενο.
- Έχει σημασία το πώς παρουσιάζουμε το περιεχόμενο;
- Ε, όσο να 'ναι έχει. Άλλο να διαβάζεις, άλλο να παρακολουθείς...
- Όμως είμαστε στο 2014 κι ο κόσμος εδώ και χρόνια παράγει περιεχόμενο με πολλούς και διαφορετικούς τρόπους!
- Συμπερασματικά, λοιπόν, το περιεχόμενο είναι περιεχόμενο είναι περιεχόμενο.

- (Αυτό το τελευταίο, μόλις και μετά βίας κρατήθηκα για να μην το γράψω τέσσερις φορές.)
- Παραμένοντας, λοιπόν, αφοσιωμένοι στην παραγωγή (ποιοτικού) περιεχομένου με διάφορους τρόπους, κάνουμε τον τελευταίο καιρό τα πρώτα βήματα για τη διεύρυνση του (αναγνωστικού) κοινού στο οποίο απευθυνόμαστε.
- Αναφέρομαι στην αγγλική εκδοχή του deltaHacker, αλλά και του deltaCast. Το νέο site που θα φιλοξενήσει το νέο εγχείρημα είναι σχεδόν έτοιμο και, σύμφωνα με όσα μου λένε οι καλοί συνεργάτες και φίλοι που έχουν αναλάβει το σχετικό πρότζεκτ, μέσα στον Απρίλιο θα μπορέσουμε να το βγάλουμε στον αέρα.
- Δεν ξέρω πώς θα κριθεί η νέα μας προσπάθεια, δεν σας κρύβω όμως ότι είμαι εξαιρετικά αισιόδοξος για το μέλλον.
- Την ίδια στιγμή, έχω λίγη αγωνία για το feedback που ελπίζω να μας δώσετε εσείς, οι καλοί φίλοι και συνδρομητές του deltaHacker.
- Ό,τι και να λέω, ό,τι και να γράφω, να ξέρετε πως θέλω να μας κρίνετε αυστηρά.
- Όπως έχει συμβεί και στο παρελθόν, έτσι και τώρα θα μας κάνει μεγάλο καλό να σας ακούσουμε.
- Και η αγγλική εκδοχή, με τη σειρά της, να ξέρετε πως θα κάνει μεγάλο καλό και στο deltaHacker.
- Σκεφτείτε, π.χ., ότι πράγματα που θα εισάγουμε και θα κριθούν θετικά από το διεθνές κοινό —ακόμα δυσκολεύομαι να συνειδητοποιήσω ότι μιλάμε για αναγνώστες απ' όλον τον πλανήτη—, θα τα φέρουμε *και* στο deltaHacker.
- Αλλά κάθε πράγμα στην ώρα του.
- Λες και είμαι από εκείνους που μπορούν να περιμένουν υπομονετικά, δηλαδή.
- Τέλος πάντων.

Καλά να περνάτε,

Ο Προκομμένος Σας (TM)



Το **deltaHacker** είναι το ένα και **μοναδικό** περιοδικό με θεματολογία το ethical **hacking**, τα **δίκτυα** και την **ασφάλεια**, τον **προγραμματισμό** και τα **ηλεκτρονικά**.
Το **deltaHacker** δεν γυρνάει εδώ κι εκεί στα περίπτερα, αλλά για τρίτη χρονιά στη σειρά **έρχεται κατευθείαν σ' εσάς**.
Αν είστε από εκείνους που δεν βλέπουν τον υπολογιστή τους ως άλλη μια οικιακή συσκευή και δεν αντιμετωπίζετε το Internet ως ένα δίκτυο που αρχίζει και τελειώνει με το Facebook και το Twitter, τότε είναι **109% βέβαιο ότι θα αγαπήσετε το deltaHacker**, καθώς κι ότι θα θέλετε την παρέα του **κάθε μήνα**.
Παραγγείλετε τώρα μία από τις συμφέρουσες συνδρομές από το
http://deltahacker.gr/pdf12order
(ηλεκτρονική έκδοση)
ή από το
http://deltahacker.gr/order
(έντυπη έκδοση, έως το τεύχος 019)
Σημείωση: Την ύλη **όλων των τευχών** που έχουν κυκλοφορήσει ως τώρα μπορείτε να τη βλέπετε στο
deltahacker.gr/category/magazine-issues



Ονέι μέιλ

Αγαπητοί μου,

Ο Niels Bohr, ένας από τους φυσικούς των οποίων οι εργασίες οδήγησαν τον Heisenberg και τον Schroedinger στην διατύπωση της Κβαντομηχανικής, αρεσκόταν να διηγείται την ακόλουθη ιστορία: Ένα αγοράκι μπαίνει σε ένα μαγαζί με μια πεντάρα στο χέρι και ρωτάει: "Μπορώ να αγοράσω ανάμικτες καραμέλες με μια πεντάρα;" Ο μαγαζάτορας παίρνει δύο καραμέλες από το βάζο, τις δίνει στο αγόρι και του λέει: "Ορίστε. Μπορείς να τις αναμίξεις μόνος σου."

Για κάποιο λόγο η ιστορία αυτή μου έχει κολλήσει από το πρωί στο μυαλό.

Οφείλω να ομολογήσω ότι σήμερα είμαι πολύ μπερδεμένος.

Κάθομαι τόσες ώρες τώρα για να ετοιμάσω τη μαρτιάτικη επιστολή μου και δεν ξέρω ούτε από πού να πρωταρχίσω ούτε σε τι να δώσω βάρος.

Πλησιάζει μεσημέρι κι εγώ συνεχίζω να κοιτάζω τα τεφτέρια μου σαν το χάνο.

Παντού γίνεται χαμός.

Τι είναι πιο σημαντικό, πείτε μου: Το ότι συνεχίζονται άκαρπες οι έρευνες για το χαμένο Μπόινγκ ή ότι στην Τουρκία ο Ερντογάν προσπαθεί να κλείσει το Twitter;

Το ότι στην Κριμαία αποφάσισαν με δημοψήφισμα να επιστρέψουν στη Ρωσική αγκαλιά, ή ότι οι βουλευτές της Χρυσής Αβγής (ΤΜ) έχουν αγχωθεί με τις προφυλακίσεις και προσπαθούν να την κάνουν με πλάγια πηδηματάκια για να σώσουν το τομάρι τους;

Το ότι κλείστηκε η συμφωνία με την Τρόικα, οπότε από στιγμή σε στιγμή εισπράττουμε τη δόση, ή ότι σήμερα είναι, και τυπικά, η πρώτη μέρα της άνοιξης;

Ότι ανιχνεύτηκαν για πρώτη φορά ενδείξεις βαρυτικών κυμάτων χρονολογούμενων από την εποχή του μπιγκ μπανγκ, ή ότι συνεχίζονται οι περιορισμοί στη διάθεση των μη συνταγογραφούμενων φαρμάκων;

Το ότι ίσως βρέθηκε ο Satoshi Nakamoto, ο εμπνευστής, υποτίθεται, του Bitcoin, ή ότι η τρόικα πιέζει για την απελευθέρωση της αγοράς υπηρεσιών κομμωτικής;

Ωχ, καθίστε, λάθος. Αυτό το τελευταίο δεν είναι για μας. Στη γαλανή μας πατρίδα προβλήματα εμείς με τα κομμωτήρια δεν έχουμε. Με τα φαρμακεία, ναι· με τους ταξιτζήδες, ναι· με το φρέσκο γάλα, ναι. Αλλά με τα κομμωτήρια, όχι. Όλα κι όλα. Σε 'μας τα κομμωτήρια είναι εντάξει. Στην Κύπρο έχουνε το πρόβλημα. Το σωστό να λέγεται.

Πλάκα πλάκα, αναλογίζομαι κάποιες στιγμές πόσο άχαρη πρέπει να είναι η δουλειά των διαπραγματευτών της τρόικας. Τους λυπάται η ψυχή μου. Οι άγγλοι έχουν την κατάλληλη λέξη γι αυτό: Unrewarding.

Αλλά πραγματικά: Σκεφτείτε τώρα να είσαι διαπραγματευτής από τη Δανία ή τη Μεγάλη Βρετανία και να προσπαθείς να εξηγήσεις σε ανθρώπους, που δεν έχουν και τόση διάθεση να καταλάβουν, ότι το να διατίθενται τα παναντόλ και τα ντεπόν στα σουπερμάρκετ δεν αποτελεί κίνδυνο για τη δημόσια υγεία, τουλάχιστον όχι μεγαλύτερο κίνδυνο από όσον αποτελεί η διάθεση ουίσκι ή τσιγάρων, και ότι, τέλος πάντων, όχι μόνο κακό δεν θα κάνει αλλά θα οδηγήσει και σε μείωση των τιμών.

Τίποτα. Οι γίγαντες των διαπραγματευτικών μας τραπεζιών δεν σηκώνουν μύγα στο σπαθί τους. Δε χαμπαριάζουν από αυτά. Η προστασία της δημόσιας υγείας πάνω από όλα.

Παράδειγμα είναι αυτό, ΟΚ; Μην ανοίξουμε συζήτηση, σας παρακαλώ. Παράδειγμα που καταδεικνύει ένα πρόβλημα, ωστόσο. Διότι είναι ηλίου φαεινότερον πως πίσω από την πρεμούρα για την προστασία της δημόσιας υγείας κρύβεται μια άλλη πρεμούρα, πολύ πιο σαφής και --δυστυχώς-- πιο εύλογη. Κρύβεται η πρεμούρα να προστατευτούν τα προνόμια των φαρμακείων. Τα περί δημόσιας υγείας είναι γαρνιτούρες για τα παράθυρα των τηλεοράσεων. Κακά τα ψέματα.

Θα μου πείτε, βέβαια: Γιατί τέτοια πρεμούρα για την προστασία των προνομίων των φαρμακείων; Τι ζόρι τραβάνε;

Εδώ είναι το μυστικό, αγαπητοί μου. Δεν τραβάνε κανένα ζόρι με τα φαρμακεία. Αν όμως κάνεις κάτι που θα θίξει πραγματικά τα φαρμακεία, θα πρέπει μετά να κάνεις και κάτι άλλο, για να θίξεις τα προνόμια των ηλεκτρολόγων. Η των αρτοπωλών. Η των δικηγόρων. Η των ταξιτζήδων. Η των γαλακτοπαραγωγών. Η των δεν-ξέρω-ποιων-άλλων.

Ανοίγεις τον ασκό του Αιόλου. Άντε βγάλε άκρη μετά. Καλύτερα να μην κάνεις τίποτα. Αρχίνα να τους λες για την δημόσια υγεία, και κάποια στιγμή θα βαρεθούν και θα φύγουν.

Αυτό ελπίζεις εσύ, βέβαια. Γιατί οι τροϊκανοί δεν μοιάζουν να βιάζονται να φύγουν. Δεν είναι στο χέρι τους, άλλωστε, ακόμα και να ήθελαν να φύγουν. Θα τρέχουν μετά από τηλεδιάσκεψη σε τηλεδιάσκεψη να εξηγούν τα ανεξήγητα στους Ευρωπαίους. Μπρος γκρεμός και πίσω ρέμα.

Γι' αυτό σας λέω. Η δουλειά του τροϊκανού δεν μου μοιάζει καθόλου αξιοζήλευτη.

Τέλος πάντων. Παρεκτρέπομαι.



Ναι, λοιπόν. Δεν ξέρω από πού να το πιάσω σήμερα.

Να, ας πούμε, μια σελίδα εδώ, στην τύχη. Τα bitcoins.

Τι να γράψω για bitcoin; Να σας γράψω ότι στο Mt. Gox, το ηλε-ανταλλακτήριο που χρεωκόπησε, βρήκανε τα 200 από τα 850 χιλιοbitcoin που είχαν χάσει, και ψάχνουν να βρούνε τα υπόλοιπα;

Ποιον απασχολεί αυτό;

Η μήπως (άλλη σελίδα) να επιστρέψω στα δικά μας, και να γράψω για τις ενδοκυβερνητικές διαφωνίες γύρω από το πώς θα διανεμηθεί το κοινωνικό μέρισμα, τώρα που πετύχαμε πρωτογενές πλεόνασμα;

Τώρα που το ξανασκέφτομαι, μήπως αυτές οι δύο ειδήσεις σχετίζονται; Γιατί, βλέπω εδώ στον γκουγκλ, πόσο έχει πάει η ισοτιμία ευρώ - bitcoin, και σκέφτομαι ότι 650 χιλιαρικάκια (bitcoin) φτάνουν μια χαρά για μέρισμα στους αναξιοπαθείς της γαλανής μας πατρίδος, ενστόλους και μη. Ακόμα και για την αποκατάσταση της μισθολογικής αδικίας σε βάρος των δικαστικών, φτάνουν.

Τέλος πάντων. Ποιόν απασχολεί το θέμα αυτό; Εμένα πάντως όχι. Δεν είμαι ούτε ένστολος, ούτε δικαστικός. Και μετανιώνω που δεν έγινα φαρμακοποιός.

Μετά κοιτάζω στα διεθνή. Στην Ουκρανία, ας πούμε. Πάνε οι Ρώσοι, οργανώνουν δημοψήφισμα, και μετά προσαρτούν την Κριμαία κι αλλάζουν τα σύνορα. Αυτό είναι σοβαρό. Πολύ σοβαρό.

Αλλά εμείς δε μασάμε. Τα καταλαβαίνουμε όλα. Βγαίνουν στις τηλεοράσεις όλοι οι "ειδικοί" ρωσολόγοι και μιλούν για αποκατάσταση μιας ιστορικής αδικίας δεκαετιών. Και δεν καταλαβαίνουν ότι οι ιστορικές αδικίες δεν έρχονται με στάμπα με την ημερομηνία παστερίωσης, όπως, ξέρω 'γω, το γάλα.

Αυτό που για τον ένα είναι εθνική επέτειος και πηγή περηφάνειας, για τον άλλο είναι όνειδος και ιστορική αδικία. Κι αν αρχίσεις να τις αποκαθιστάς, ούτε φίδι στον κόρφο σου. Δεν θα ξέρεις από πού σου ήρθε.

Να μου το θυμηθείτε.

Τέλος πάντων. Αυτά που λέτε, αγαπητοί μου. Από το πρωί έτσι.

Αφού με είδε κάποια στιγμή να φυσάω και να ξεφυσάω και να χασμουριέμαι, και η μητέρα της εξαδέλφης --επιστρέφοντας από την λαϊκή φορτωμένη με ψώνια-- έβγαλε διάγνωση: "Είσαι ματιασμένος."

"Μα τι λέτε, ματιασμένος; Ποιος να με ματιάσει; Γιατί;"

"Δεν ξέρω. Αυτά να τα ψάξεις εσύ που ξέρεις με ποιους κάνεις παρέα", είπε αποφασιστικά η μητέρα της εξαδέλφης, που είναι Ινδουίστρια. "Είσαι ματιασμένος και παίρνω τώρα την Κυρία Καίτη, να έρθει να σε ξεματιάσει."

Έτσι που λέτε λοιπόν. Έξω χαρά θεού, ο ήλιος λάμπει, τα λουλούδια ανθίζουν, τα πουλιά λαλούν, κι εμείς εδώ μέσα κλεισμένοι, να περιμένουμε την Κυρία Καίτη να έρθει να με ξεματιάσει.

Από στιγμή σε στιγμή θα είναι εδώ.

"Μήπως να της πούμε να μας φέρει και λίγες ανάμικτες καραμέλες;" ακούστηκε από την κουζίνα η εξαδέλφη. "Πλάκα θα έχει."

Σας ασπάζομαι,
Θείος Ακάκιος


Τα επεισόδια του deltaCast, μοιάζουν με ταινίες.
Στη διάρκεια -- όχι στην ποιότητα.
deltaCast s02e02
notMe-CryptoBSD
http://deltahacker.gr/category/deltacast

Skill: Advanced
Tags: VMware, Assembly, IDTR, anti-debugging, anti-disassembling

anti-VM

by George of the Jungle

Τεχνικές anti-virtualization!

Τα δικά μας προγράμματα έχουν αξιοπρέπεια. Έχουν ένα επίπεδο, βρε αδερφέ! Όταν εκτελούνται σε πραγματικά συστήματα, λειτουργούν κανονικά. Αντίθετα, όταν αντιλαμβάνονται ότι βρίσκονται μέσα σε κάποιο εικονικό περιβάλλον, αρνούνται πεισματικά να ξεκινήσουν. Αναρωτιέστε πώς μπορεί να γίνει κάτι τέτοιο; Στο παρόν άρθρο θα μελετήσουμε τη λογική και τις τεχνικές υλοποίησης μιας τέτοιας δικλείδας ασφάλειας...

Η τεχνολογία των εικονικών μηχανών χαίρει ιδιαίτερης εκτίμησης από αναλυτές malware, pen-testers, network engineers, προγραμματιστές κ.ά. Μέσα σε ένα εικονικό σύστημα μπορεί κανείς να διεξάγει κρίσιμα πειράματα, χωρίς να θέτει σε κίνδυνο τον πραγματικό υπολογιστή του. Για παράδειγμα, αν θέλει κανείς να τσεκάρει τη συμπεριφορά μιας εφαρμογής ώστε να διαπιστώσει το αν λειτουργεί κακόβουλα ή όχι, τότε μπορεί —για να μην πούμε ότι επιβάλλεται— να την τρέξει σε ένα εικονικό σύστημα. Αν προκληθεί ζημιά, θα περιορίζεται στο εικονικό σύστημα και θα διορθώνεται πανεύκολα, με τη βοήθεια κάποιου snapshot. Βέβαια, ο συγκεκριμένος τρόπος εργασίας δεν οδηγεί πάντα σε ασφαλή συμπεράσματα. Ένα κακόβουλο πρόγραμμα ενδέχεται να αντιληφθεί ότι λειτουργεί εντός εικονικού περιβάλλοντος και να επιλέξει να κρύψει την επικίνδυνη συμπεριφορά του! Αυτού του είδους η ευφυΐα είναι αρκετά επικίνδυνη όταν την έχει κάποιο malware, αλλά υπάρχουν και περιπτώσεις στις οποίες μπορεί να αποδειχθεί χρήσιμη — ακόμα και σωτήρια! Για παράδειγμα, σκεφτείτε το σενάριο κατά το οποίο θέλουμε να προστατέψουμε ένα πρόγραμμα από τη μη εξουσιοδοτημένη τροποποίηση (cracking). Σε αυτή την περίπτωση, δεν θα θέλουμε να εκτελείται μέσα σε εικονικά συστήματα. Βλέπετε, όσοι ασχολούνται με το Reverse Engineering συνηθίζουν να εργάζονται σε virtual machines, ώστε να αποφεύγουν τις παγίδες και να ξεπερνούν τα διάφορα anti-debugging tricks.

Στο παρόν άρθρο θα εξετάσουμε τις τεχνικές με τις οποίες μπορεί να εξασφαλίσει κανείς ότι το πρόγραμμά του θα αρνείται να λειτουργήσει όποτε αντιλαμβάνεται ότι εκτελείται μέσα σε μία εικονική μηχανή. Σε γενικές γραμμές, κάτι τέτοιο επιτυγχάνεται με την προσθήκη του κατάλληλου κώδικα, πριν ακόμα κι από το main initialization του προγράμματος. Ο κώδικας που θα εξετάσουμε στο άρθρο θα ελέγχει, με διάφορους τρόπους, για το αν βρισκόμαστε σε πραγματικό σύστημα ή σε εικονική μηχανή. Στη δεύτερη περίπτωση, ένα φιλικό μήνυμα θα ενημερώνει το χρήστη ότι διαπιστώθηκε λειτουργία εντός virtual machine και το πρόγραμμα θα τερματίζεται. Όπως αντιλαμβάνεστε, το μήνυμα θα είναι φιλικό, μιας και πρόκειται για κώδικα που γράψαμε εμείς και προορίζεται για τις ανάγκες του άρθρου. Στην πραγματικότητα, θα μπορούσε να είναι κι ένα σκέτο "FATAL EXIT" ;)

Μια στιγμή, κύριοι...

Κάπου εδώ σάς ακούμε να διαμαρτύρεστε, ισχυριζόμενοι ότι ένας έλεγχος μπορεί, σε γενικές γραμμές, να ξεπεραστεί αρκετά εύκολα. Στο κάτω-κάτω, όσο περίπλοκος κι αν είναι, καταλήγει πάντα σε ένα "if statement". Όταν το πρόγραμμα μεταφράζεται σε Assembly, η συγκεκριμένη δομή ελέγχου μετατρέπεται σε μια εντολή του τύπου JNZ (Jump if Not Zero) ή JZ (Jump if Zero). Επομένως, ένας χρήστης που θέλει να αντιστρέψει το συγκεκριμένο έλεγχο, μπορεί απλά να αλλάξει το JNZ σε JZ ή το ανάποδο (ανάλογα με το ποια εντολή έχει χρησιμοποιηθεί). Με αυτή την απλή επέμβαση, μπορεί να εξασφαλίσει ότι το πρόγραμμά μας θα τρέχει σε εικονικές μηχανές, χωρίς κανένα πρόβλημα. Έτσι δεν είναι; Ε, λοιπόν, όχι. Δεν είναι έτσι. Ωστόσο, το αντίμετρο για μια τέτοια επέμβαση θα το δούμε στο τέλος του άρθρου. Για την ώρα θα εξετάσουμε τη λογική και την υλοποίηση του ελέγχου, για το αν βρισκόμαστε μέσα σε κάποιο εικονικό περιβάλλον.

Απαραίτητες παραδοχές

Πριν ξεκινήσουμε οφείλουμε να πούμε δύο λόγια για το πώς θα εργαστούμε, καθώς και για τα εργαλεία που θα χρησιμοποιήσουμε. Κατ' αρχάς, για τις ανάγκες του άρθρου στηριχτήκαμε στην εφαρμογή VMware Workstation 10 (trial edition,

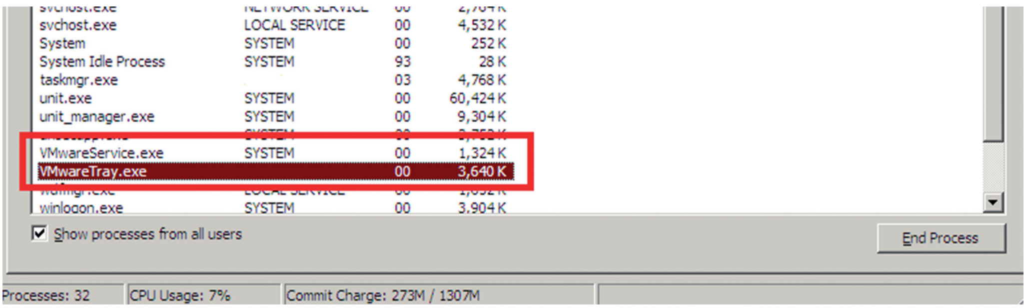
www.vmware.com). Για τη συγγραφή του κώδικα στραφήκαμε στο περιβάλλον C++ Builder XE4, της εταιρείας Embarcadero (πάλαι ποτέ κραταιά Borland). Επιλέξαμε το συγκεκριμένο περιβάλλον επειδή τα εκτελέσιμα που παράγει είναι απαλλαγμένα από το ενοχλητικό (για σημαντική μερίδα ανθρώπων) dot-NET Framework της Microsoft. Μπορείτε να κατεβάσετε μια δοκιμαστική εκδοχή του συγκεκριμένου περιβάλλοντος από εδώ: www.embarcadero.com/downloads. Για τις δοκιμές μας στήσαμε δύο εικονικές μηχανές. Στη μία εγκαταστήσαμε τα Windows XP Professional SP2, ενώ στην άλλη τα Windows 8 Professional. Χρησιμοποιήσαμε δύο λειτουργικά συστήματα, για να έχουμε όσο το δυνατόν πληρέστερη εικόνα. Τέλος, σημειώστε ότι μπορείτε να κατεβάσετε ολόκληρο τον πηγαίο κώδικα που θα εξετάσουμε από εδώ http://bit.ly/dh030avmlibexe. Επίσης, αν θέλετε να πάρετε μια γεύση από όσα θα πετύχουμε, μπορείτε να κατεβάσετε και το εκτελέσιμο που προκύπτει μεταγλωττίζοντας τον κώδικά μας, από το http://bit.ly/dh030avmsrc.

Γενικές κατευθύνσεις

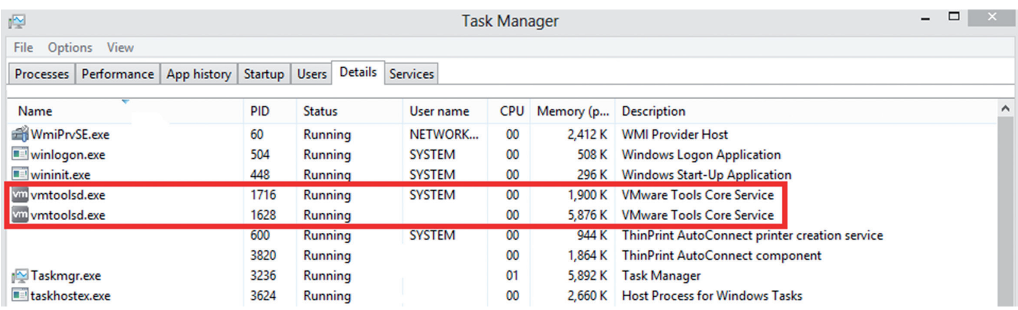
Ο κώδικας που θέλουμε να γράψουμε θα αναζητά αποδεικτικά στοιχεία που μαρτυρούν τη λειτουργία μέσα σε εικονικό περιβάλλον. Στη συνέχεια, ανάλογα με το αν εντοπίστηκαν τέτοια στοιχεία ή όχι, θα τυπώνει το κατάλληλο μήνυμα και θα συνεχίζει ή θα διακόπτει την εκτέλεση του προγράμματος. Όμως ποια είναι εκείνα τα στοιχεία που φανερώνουν την πραγματική φύση του συστήματος; Πού πρέπει να ψάξει ο κώδικάς μας και τι ακριβώς πρέπει ν' αναζητήσει; Αναφέρουμε στη συνέχεια τρεις βασικές μεθόδους/περιοχές έρευνας.

- Στις εικονικές μηχανές λειτουργούν συνήθως ορισμένες διεργασίες, οι οποίες διευκολύνουν την επικοινωνία μεταξύ του εικονικού και του φυσικού συστήματος, ενώ βελτιώνουν και την εμπειρία χρήσης (π.χ., VMware Tools ή VirtualBox Guest Additions). Πρόκειται για διεργασίες που δεν απαντώνται ποτέ σε κάποιο πραγματικό σύστημα. Επομένως, αν ο κώδικας εντοπίσει κάποια από αυτές, μπορεί να συμπεράνει με ασφάλεια ότι εκτελείται μέσα σε εικονική μηχανή.
- Σε ένα πραγματικό σύστημα, τα εξαρτήματα του υπολογιστή έχουν το όνομα που τους έδωσε ο κατασκευαστής τους. Για παράδειγμα, στις πληροφορίες για έναν σκληρό δίσκο θα συναντήσουμε κάπου το όνομα του κατασκευαστή, καθώς και το μοντέλο της συσκευής (π.χ., Western Digital Green Power 500). Στις εικονικές μηχανές τα πράγματα είναι πολύ διαφορετικά. Τα ονόματα των συσκευών που απαρτίζουν το σύστημα όχι μόνο φανερώνουν ότι πρόκειται για εικονική συσκευή, αλλά δηλώνουν και την υλοποίηση (π.χ., VMware, VirtualBox κ.ά.). Φυσικά, ο χρήστης μιας εικονικής μηχανής μπορεί να τροποποιήσει αυτά τα ονόματα και να επιλέξει κάποια που να παραπέμπουν σε πραγματικό σύστημα. Ωστόσο, είναι ελάχιστοι αυτοί που θα μπουν στον κόπο να κάνουν κάτι τέτοιο. Επομένως, ο κώδικάς μας θα μπορούσε να τσεκάρει και τα ονόματα μερικών συσκευών.
- Μια τρίτη μέθοδος περιλαμβάνει τμήματα κώδικα που έχουν γραφτεί σε Assembly, αποτελούν πραγματικά αριστουργήματα και προέρχονται από την παγκόσμια κοινότητα των προγραμματιστών. Πρόκειται για ρουτίνες που εξαπατούν το εκάστοτε σύστημα και το υποχρεώνουν να αποκαλύψει την πραγματική του υπόσταση (εικονική ή φυσική). Μερικά τέτοια τμήματα κώδικα θα εξετάσουμε στο 3ο και τελευταίο τμήμα του άρθρου, όπου και θα μπορέσουμε να πούμε περισσότερα για αυτά τα μικρά διαμάντια κώδικα.

Όπως αντιλαμβάνεστε, οι πρώτες δύο μέθοδοι έρευνας είναι αρκετά απλές στην υλοποίηση. Στην μία αρκεί να πάρουμε μια λίστα με τις ενεργές διεργασίες (active processes enumeration) και να τσεκάρουμε τα στοιχεία της, ενώ στην άλλη αρκεί να εξετάσουμε τα ονόματα των συσκευών, είτε από το registry των Windows είτε από τον αντίστοιχο μηχανισμό του εκάστοτε λειτουργικού συστήματος. Σημειώστε ότι αν κι αυτές οι μέθοδοι δεν είναι καθόλου εντυπωσιακές, θεωρούνται ως οι πλέον αξιόπιστες. Βλέπετε, αν εντοπίσουμε κάποια διεργασία ή κάποια συσκευή που παραπέμπει σε εικονική μηχανή, το συμπέρασμα θα είναι απόλυτα σίγουρο και δεν θα χωράει καμία αμφισβήτηση.



Εικόνα 1
Έχουμε φορτώσει τον TaskManager στα WindowsXP κι έχουμε εντοπίσει δύο διεργασίες που παραπέμπουν αμέσως στο VMware!



Εικόνα 2
Στα Windows 8 έχουμε εντοπίσει την ενεργή διεργασία vmtoolsd.exe. Αυτό σημαίνει ότι βρισκόμαστε σε εικονική μηχανή και, μάλιστα, ότι έχουν εγκατασταθεί και ξεκινήσει τα VMware Tools.

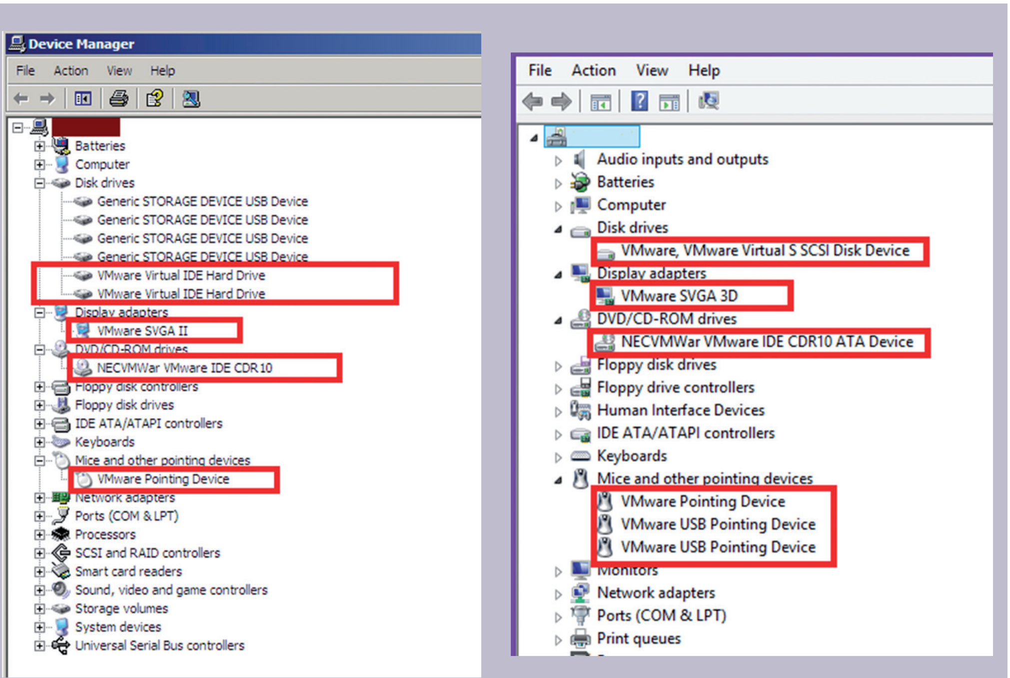
Έρευνα στο λογισμικό

Στις εικόνες 1 και 2 βλέπουμε μερικές από τις διεργασίες που τρέχουν σε ένα σύστημα με τα Windows XP και τα Windows 8 Professional, αντίστοιχα. Και στις δύο περιπτώσεις διαπιστώνουμε ότι εκτελούνται ορισμένες διεργασίες που μαρτυρούν ότι πρόκειται για εικονική μηχανή του VMware. Επομένως, στην αρχή του προγράμματός μας θα μπορούσαμε να ενσωματώσουμε έναν έλεγχο για την παρουσία κάποιας από τις εν λόγω διεργασίες. Ο κώδικας που υλοποιεί το παραπάνω σκεπτικό θα μπορούσε να γραφεί ως εξής:

```
HANDLE hSnapShot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,0);
PROCESSENTRY32* processInfo = new PROCESSENTRY32;
processInfo->dwSize = sizeof(PROCESSENTRY32);
```

```
bool VMProcessIsFound = false;

while(Process32Next(hSnapShot,processInfo)!=FALSE) {
    if (wcscmp (L"vmtoolsd.exe", processInfo->szExeFile) == 0 ||
    wcscmp (L"VMwareService.exe", processInfo->szExeFile) == 0 ||
    wcscmp (L"VMwareTray.exe", processInfo->szExeFile ) == 0) {
        VMProcessIsFound=true;
    }
}
if (VMProcessIsFound) {
    MessageBox(NULL, L"Είμαστε μέσα στο matrix!", L"Report", MB_
ICONINFORMATION);
} else {
    MessageBox(NULL, L"Βρισκόμαστε σε πραγματικό σύστημα!", L"Report",
MB_ICONINFORMATION);
}
CloseHandle(hSnapShot);
```



Εικόνες 3 & 4
Ο Device manager περιέχει ολόκληρη τη λίστα με τις συσκευές του συστήματος. Τόσο στα Windows XP, όσο και στα Windows 8, υπάρχουν πολλές συσκευές που στην περιγραφή τους περιέχουν τη λέξη "VMware".

Στις πρώτες γραμμές προετοιμάζουμε το έδαφος, ώστε να προβούμε σε ένα active process enumeration. Στη συνέχεια, με τη δομή επανάληψης while διαβάζουμε το όνομα καθεμίας από τις ενεργές διεργασίες. Αν συναντήσουμε ένα από τα επικηρυγμένα ονόματα (vmtoolsd.exe, VMwareService.exe ή VMwareTray.exe), τότε είμαστε βέβαιοι ότι ο κώδικας εκτελείται μέσα σε εικονική μηχανή. Τι θα γίνει, όμως, αν οι ζητούμενες διεργασίες είναι ανενεργές; Ο κώδικάς μας θα αποτύχει να τις εντοπίσει και θα θεωρήσει ότι εκτελείται σε πραγματικό σύστημα. Για να αποφύγουμε αυτό το πρόβλημα, μπορούμε να επεκτείνουμε τη σχετική αναζήτηση στο registry. Για παράδειγμα, μπορούμε να τσεκάρουμε τις εγκατεστημένες εφαρμογές ή εκείνες που ξεκινούν αυτόματα με το σύστημα. Όπως καταλαβαίνετε, ακόμα κι αν οι εφαρμογές που αναζητάμε δεν εκτελούνται, είναι βέβαιο ότι θα έχουν αφήσει τα ίχνη τους στο μητρώο.

Έρευνα στο υλικό

Τώρα μπορούμε να προχωρήσουμε στη δεύτερη μέθοδο και να τσεκάρουμε τα ονόματα των συσκευών. Για αυτή την αναζήτηση θα κοιτάξουμε — που αλλού; — στο μητρώο των Windows. Αναρωτιέστε για το αν υπάρχουν εκεί μέσα τα ονόματα των συσκευών; Φυσικά και υπάρχουν! Από εκεί τα διαβάζει άλλωστε και ο Device Manager των Windows.

Ρίξτε μια ματιά στις εικόνες 003 και 004, για να καταλάβετε τι εννοούμε όταν λέμε ότι θα τσεκάρουμε τα ονόματα των συσκευών. Στην πρώτη εικόνα φαίνεται ο Device Manager του εικονικού συστήματος με τα Windows XP, ενώ στη δεύτερη φαίνεται η αντίστοιχη περιοχή του εικονικού συστήματος με τα Windows 8. Τι μπορεί να διαπιστώσει κανείς εύκολα; Υπάρχουν συσκευές που το όνομα τους περιέχει την "κωδική" λέξη VMware. Αυτά είναι τα αποδεικτικά στοιχεία που αναζητάμε. Με μια πολύ απλή αναζήτηση της λέξης "VMware" στο μητρώο (μέσα από το γνωστό εργαλείο των Windows ονόματι regedit) παίρνουμε τα αποτελέσματα που φαίνονται στους ακόλουθους πίνακες…

Αποτελέσματα αναζήτησης στο μητρώο των Windows XP:

```
HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target
Id 0\Logical Unit Id 0\
Identifier : VMware Virtual IDE Hard Drive
```

```
HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\Scsi\Scsi Port 1\Scsi Bus 0\Target
Id 0\Logical Unit Id 0\
Identifier : NECVMWar VMware IDE CDR10
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4D36E968-E325-
11CE-BFC1-08002BE10318}\0000\Settings
Device Description : VMware SVGA II
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4D36E96F-E325-
11CE-BFC1-08002BE10318}\0000

DriverDesc : VMware Pointing Device

ProviderName : VMware, Inc.
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{53f56307-
b6bf-11d0-94f2-00a0c91efb8b}\##?#IDE#DiskVMware_Virtual_IDE_Hard_Drive_
__________00000001#3030303030303030303030303030303030303130#{53f56307-
b6bf-11d0-94f2-00a0c91efb8b}

DeviceInstance : IDE\DiskVMware_Virtual_IDE_Hard_Dri
ve___________00000001\3030303030303030303030303030303030303130
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{53f56308-
b6bf-11d0-94f2-00a0c91efb8b}\##?#IDE#CdRomNECVMWar_VMware_IDE_CDR10____
___________1.00____#3031303030303030303030303030303030303130#{53f56308-
b6bf-11d0-94f2-00a0c91efb8b}

DeviceInstance : IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________1.00__
__\3031303030303030303030303030303030303130
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Video\{93C9713A-B7FE-
49CD-A83F-DE70957523F5}\0000

Device Description : VMware SVGA II
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\IDE\CdRomNECVMWar_VMware_
IDE_CDR10_______________1.00____\30313030303030303030303030303030303031
30

FriendlyName : NECVMWar VMware IDE CDR10

HardwareID :

IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________1.00____

IDE\NECVMWar_VMware_IDE_CDR10_______________1.00____

IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________

NECVMWar_VMware_IDE_CDR10_______________1.00____

GenCdRom
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\PCI\
VEN_15AD&DEV_0405&SUBSYS_040515AD&REV_00\3&61aaa01&0&78

DeviceDesc : VMware SVGA II

Mfg : VMware, Inc.
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\USB\Vid_0e0f&Pid_0002\5&1f8
fd7d0&0&1
LocationInformation : VMware Virtual USB Hub
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Cdrom\Enum
0 : IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________1.00____\3031303030
303030303030303030303030303130
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Disk\Enum
0 : IDE\DiskVMware_Virtual_IDE_Hard_Drive
___________00000001\3030303030303030303030303030303030303130
1 : IDE\DiskVMware_Virtual_IDE_Hard_Drive
___________00000001\3130303030303030303030303030303030303130
```

Αποτελέσματα αναζήτησης στο μητρώο των Windows 8:

```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\BIOS
SystemManufacturer : VMware, Inc.
SystemProductName : VMware Virtual Platform
```

```
HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target
Id 0\Logical Unit Id 0
Identifier : VMware, VMware Virtual S1.0
```

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winsat
PrimaryAdapterString : VMware SVGA 3D
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e968-e325-
11ce-bfc1-08002be10318}\0000
Device Description : VMware SVGA 3D
HardwareInformation.AdapterString : VMware SVGA 3D
ProviderName : VMware, Inc.
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e96f-e325-
11ce-bfc1-08002be10318}\0000
DriverDesc : VMware Pointing Device
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e97d-e325-11ce-bfc1-08002be10318}\0027
HardwareInformation.BiosString : VMware virtual machine
HardwareInformation.DacType : VMware, Inc
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Class\{4d36e97d-e325-11ce-bfc1-08002be10318}\0132
DriverDesc : VMware VMCI Bus Device
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{53f56307-b6bf-11d0-94f2-00a0c91efb8b}\##?#SCSI#Disk&Ven_VMware_&Prod_VMware_Virtual_S#5&1982005&0&000000#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
DeviceInstance : SCSI\Disk&Ven_VMware_&Prod_VMware_Virtual_S\5&1982005&0&000000
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses\{53f56308-b6bf-11d0-94f2-00a0c91efb8b}\##?#IDE#CdRomNECVMWar_VMware_IDE_CDR10_______________1.00____#5&290fd3ab&0&1.0.0#{53f56308-b6bf-11d0-94f2-00a0c91efb8b}
DeviceInstance : IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________1.00____\5&290fd3ab&0&1.0.0
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\SystemInformation
SystemManufacturer : VMware, Inc.
SystemProductName : VMware Virtual Platform
```

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\IDE\CdRomNECVMWar_VMware_IDE_CDR10_______________1.00____\5&290fd3ab&0&1.0.0
FriendlyName : NECVMWar VMware IDE CDR10 ATA Device
```

Όπως βλέπετε, το μητρώο των Windows περιέχει δεκάδες φορές τη λέξη "VMware". Φυσικά, η ρουτίνα που θα κατασκευάσουμε δεν χρειάζεται να αναζητά όλες τις εμφανίσεις της λέξης. Αρκεί να αναζητήσει μία ή δύο εγγραφές, οι οποίες γνωρίζουμε ότι εμφανίζονται σε κάθε εικονικό σύστημα. Για παράδειγμα, μπορούμε να γράψουμε μια ρουτίνα που θα τσεκάρει το serial number του σκληρού δίσκου και το όνομα του κατασκευαστή της κάρτας γραφικών.

Με την ευκαιρία, σημειώστε ότι το «Low Level Serial Number» (ή απλά LLSN) ενός δίσκου είναι μοναδικό και εντελώς διαφορετικό από το Volume Serial Number. Το LLSN δεν επιδέχεται καμία τροποποίηση, ενώ για το δεύτερο υπάρχουν διάφορα εργαλεία που επιτρέπουν να το αλλάξουμε[1]. Επειδή το LLSN είναι κάτι μοναδικό, μπο-

1. Δείτε, για παράδειγμα, εδώ: http://bit.ly/cshddsnvid

ρεί κανείς να στηριχτεί σε αυτό ώστε να εξασφαλίσει ότι η εφαρμογή του θα τρέχει *μόνο* σε ένα μηχάνημα: Σε αυτό που έχει το σκληρό δίσκο με το συγκεκριμένο LLSN! Ας επιστρέψουμε στο θέμα, όμως.

Στη συνέχεια θα δούμε μια ρουτίνα που ελέγχει αν ο πρώτος δίσκος του συστήματος έχει το αναγνωριστικό "VMware, VMware Virtual S1.0". Συγκεκριμένα, ανατρέχει στη διαδρομή HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\Scsi\Scsi Port 0\Scsi Bus 0\Target Id 0\Logical Unit Id 0 του μητρώου κι εκεί αναζητά το subkey με όνομα "string Identifier". Η τιμή αυτού του κλειδιού, αν ο δίσκος είναι εικονικός κι έχει κατασκευαστεί με το VMware, έχει σαν τιμή το string που αναφέραμε προηγουμένως. Ακολουθεί ο κώδικας:

```
HKEY keyHandle;
char rgValue [1024];
char fnlRes [1024];
DWORD size1;
DWORD Type;

if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,L"HARDWARE\\DEVICEMAP\\Scsi\\Scsi
Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0\\",0,KEY_QUERY_
VALUE,&keyHandle)== ERROR_SUCCESS) {

   size1=1023;
   RegQueryValueEx(keyHandle, L"Identifier", NULL,&Type, (LPBYTE)
rgValue,&size1);

   MessageBox(NULL, (wchar_t*)rgValue, L"Identifier", MB_SYSTEMMODAL|MB_
ICONINFORMATION);

   if (wcscmp((wchar_t*)rgValue,L"VMware, VMware Virtual S1.0 ") == 0) {
      MessageBox(NULL,L"Virtual Machine Detected!",L"Report",MB_
ICONINFORMATION);
   } else {
      MessageBox(NULL, L"Virtual Machine not Detected!", L"Report",MB_
ICONINFORMATION);

   }

} else {
   RegCloseKey(keyHandle);

   MessageBox(NULL, L"Couldn't access Registry path!", L"Report", MB_
SYSTEMMODAL|MB_ICONINFORMATION);

}
```

Στην εικόνα 5 φαίνεται τμήμα του source ενός γνωστού Trojan Horse. Όπως βλέπετε, ο κώδικας προσπαθεί να ανιχνεύσει το αν εκτελείται σε εικονική μηχανή του VMware, εξετάζοντας τα χαρακτηριστική των συσκευών του συστήματος.

Ανίχνευση από την κάρτα δικτύου

Μιλώντας για τα χαρακτηριστικά του υλικού που μπορούν να προδώσουν μια εικονική μηχανή, θα αποτελούσε σημαντική παράλειψη να μην πούμε τίποτα για την MAC

```
loc_42AF53:                             ; CODE XREF: check_for_VMware+39↑j
push    dword ptr [eax]
call    char_upper_and_calc_hash
pop     ecx
cmp     eax, 2FE483F3h                  ; vmscsi.sys
jz      short loc_42AFB5
cmp     eax, 2FD5F8F3h                  ; vmhgfs.sys
jz      short loc_42AFB5
cmp     eax, 0CFF129A8h                 ; vmx_svga.sys
jz      short loc_42AFB5
cmp     eax, 2FDB60F3h                  ; vmxnet.sys
jz      short loc_42AFB5
cmp     eax, 2FFC2FB4h                  ; vmmouse.sys
jz      short loc_42AFB5
cmp     eax, 2FF91C94h                  ; vmdebug.sys
jz      short loc_42AFB5
inc     esi
cmp     esi, [ebp+var_8]
jb      short loc_42AF49

loc_42AF8B:                             ; CODE XREF: check_for_VMware+32↑j
push    9F408DC4h                       ; VMwareTray.exe
call    check_process_by_hash
pop     ecx
test    eax, eax
jnz     short loc_42AFBD
push    9F5784C4h                       ; VMwareUser.exe
call    check_process_by_hash
pop     ecx
test    eax, eax
```

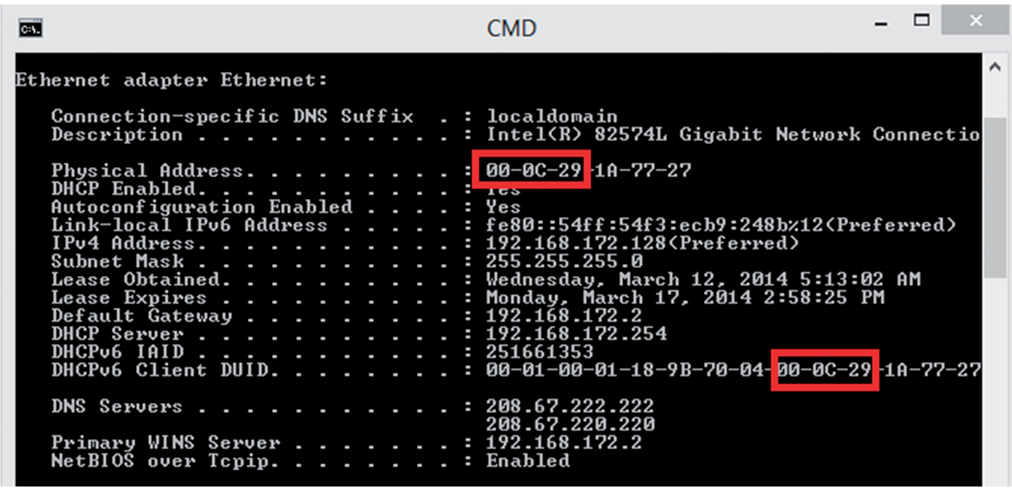
Εικόνα 5
Ο πηγαίος κώδικας ενός γνωστού Trojan, ελέγχει αν εκτελείται μέσα σε Virtual Machine του VMware.

address των καρτών δικτύου. Βλέπετε, τα πρώτα έξι bytes αυτού του μεγέθους είναι πάντοτε ίδια, σε όλες τις κάρτες γραφικών μιας εικονικής μηχανής. Συγκεκριμένα, όταν η MAC address ξεκινά με την τιμή "00-0C-29", γνωρίζουμε ότι ανήκει σε μια εικονική κάρτα δικτύου η οποία με τη σειρά της ανήκει σε ένα εικονικό σύστημα. Αν δώσουμε ipconfig /all στη γραμμή εντολών του VM με τα Windows 8, θα πάρουμε τα αποτελέσματα που φαίνονται στην εικόνα 6. Ας δούμε τώρα και τις κάρτες δικτύου στο σύστημα του γράφοντα. Τα αποτελέσματα από την εκτέλεση του ipconfig φαίνονται στην εικόνα 7. Αν τσεκάρετε τις διευθύνσεις MAC, θα διαπιστώσετε ότι καμία δεν ξεκινά με τα έξι "ειδικά ψηφία", αν και η μία κάρτα δικτύου έχει δημιουργηθεί από το VMware. Αυτό συμβαίνει διότι το σύστημα του γράφοντα είναι φυσικό κι όχι εικονικό :)

Τώρα, μπορούμε να εξετάσουμε μια συνάρτηση που αξιοποιεί την παραπάνω γνώση, για να τσεκάρει αν βρισκόμαστε σε εικονικό περιβάλλον ή όχι. Η συνάρτηση που γράψαμε επιστρέφει μια τιμή boolean (true ή false), ανάλογα με το αν η MAC address ξεκινά με τα ψηφία "00-0C-29" ή όχι:

```
bool Get_MAC_Address() {
   PIP_ADAPTER_INFO AdapterInfo;
   DWORD dwBufLen = sizeof(AdapterInfo);
   char *mac_addr = (char*)malloc(17);
   bool NICinsideVM = false;
   AdapterInfo = (IP_ADAPTER_INFO *) malloc(sizeof(IP_ADAPTER_INFO));
   if (AdapterInfo == NULL) {
      MessageBox(NULL, L"Error allocating memory needed to call 
GetAdaptersinfo\n", L"Report", MB_ICONINFORMATION);
      NICinsideVM=false;
   }
   if (GetAdaptersInfo(AdapterInfo, &dwBufLen) == ERROR_BUFFER_OVERFLOW) 
{
      AdapterInfo = (IP_ADAPTER_INFO *) malloc(dwBufLen);
      if (AdapterInfo == NULL) {
         MessageBox(NULL, L"Error allocating memory needed to call 
GetAdaptersinfo\n", L"Report", MB_ICONINFORMATION);
         NICinsideVM=false;
      }
   }
   if (GetAdaptersInfo(AdapterInfo, &dwBufLen) == NO_ERROR) {
      PIP_ADAPTER_INFO pAdapterInfo = AdapterInfo;
      do {
         sprintf(mac_addr, "%02X:%02X:%02X:%02X:%02X:%02X",
         pAdapterInfo->Address[0], pAdapterInfo->Address[1],
         pAdapterInfo->Address[2], pAdapterInfo->Address[3],
         pAdapterInfo->Address[4], pAdapterInfo->Address[5]);
         std::string str1 = mac_addr;
         std::string str2 = str1.substr(0,8);
         const char *final = str2.c_str();
         MessageBox(NULL, GetWC(final), L"Mac Address", MB_
SYSTEMMODAL|MB_ICONINFORMATION);
         if (strcmp(final,"00:0C:29")==0) {
            MessageBox(NULL, L"This is a virtual network card inside a 
VM!", L"Virtual Mac Address", MB_SYSTEMMODAL|MB_ICONINFORMATION);
            NICinsideVM = true;
         }
         pAdapterInfo = pAdapterInfo->Next;
      } while(pAdapterInfo);
   }
   free(AdapterInfo);
   if (NICinsideVM) { return true; }
}
```

Σε ένα οποιοδήποτε σύστημα, είτε πρόκειται για εικονικό είτε για φυσικό, είναι απίθανο να συναντήσουμε μόνο μια κάρτα δικτύου. Γι' αυτό το λόγο, η συνάρτησή μας παίρνει μια λίστα με όλες τις κάρτες δικτύου κι ενσωματώνει έναν βρόχο do... while, στον οποίο ελέγχει τη διεύθυνση MAC της καθεμιάς. Αν εντοπιστεί το μέγεθος "00:0C:29", η μεταβλητή NICinsideVM παίρνει την τιμή true. Όπως καταλαβαίνετε, η τιμή αυτής της μεταβλητής χρησιμοποιείται στον κύριο κορμό του προγράμματος, ώστε να ληφθεί η απόφαση για το αν θα συνεχιστεί η εκτέλεση του προγράμματος ή όχι.



Εικόνα 6
Μόλις εκτελέσαμε στο command prompt το "ipconfig /all". Όπως φαίνεται στην εικόνα, η κάρτα δικτύου έχει MAC address που ξεκινά με την τιμή "00:0C:29". Αυτό σημαίνει ότι πρόκειται για μια εικονική κάρτα δικτύου, σε μια εικονική μηχανή του VMware.

Όταν όλα τ' άλλα αποτυγχάνουν...

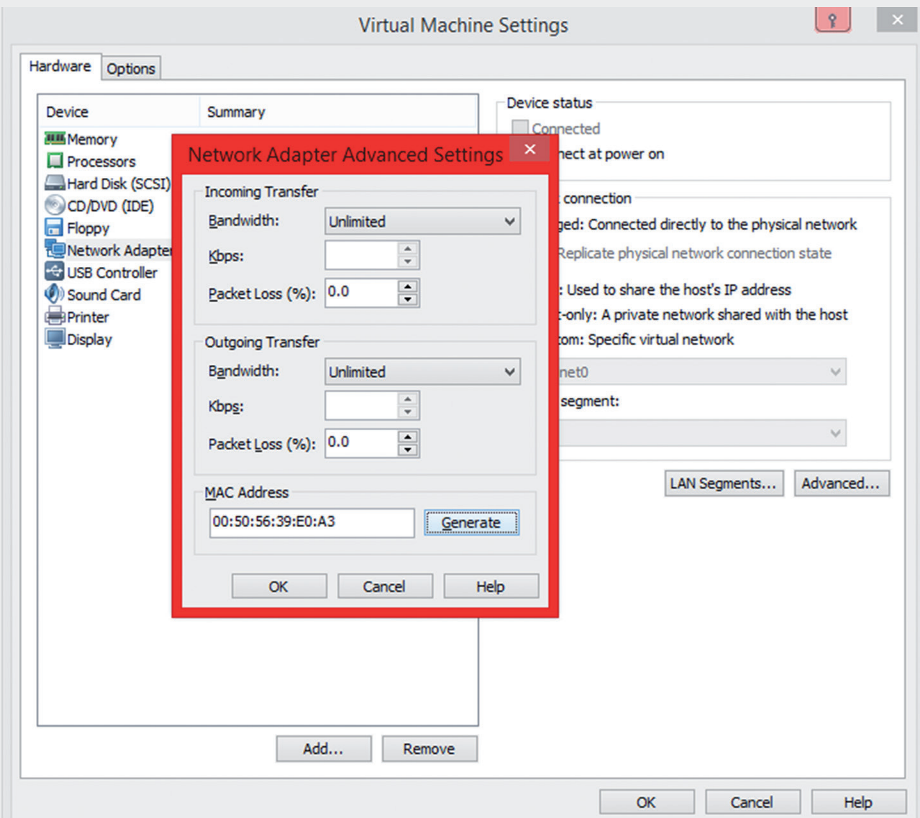
Όλα όσα είδαμε ως τώρα είναι ωραία, απλά κι αποτελεσματικά. Έχουν όμως και δύο μειονεκτήματα, που είμαστε σίγουροι ότι έχετε σκεφτεί κι εσείς. Καταρχάς, τα αποδεικτικά στοιχεία που παρουσιάσαμε εντοπίζονται μόνο σε εικονικές μηχανές του VMware. Αυτό όμως δεν είναι και τόσο και μεγάλο πρόβλημα, καθώς όλα τα προγράμματα Virtualization αφήνουν παρόμοια αποτυπώματα: Όλο και κάποιο αναγνωρίσιμο πρόγραμμα θα φυτεύουν στο guest OS, ενώ είναι βέβαιο ότι για την ονομασία των εικονικών συσκευών θα χρησιμοποιούν το δικό τους όνομα (έστω ως συνθετικό). Το άλλο μειονέκτημα όσων παρουσιάσαμε είναι πιο ζόρικο: Τι γίνεται αν ο χρήστης έχει επέμβει στις ρυθμίσεις της εικονικής μηχανής κι έχει απομακρύνει τα στοιχεία που φανερώνουν τη φύση της; Όπως βλέπετε στην εικόνα 8, το VMware επιτρέπει τον καθορισμό μιας οποιασδήποτε τιμής για τη διεύθυνση MAC της εικονικής κάρτας δικτύου. Την ίδια δυνατότητα προσφέρουν όλα τα προγράμματα Virtualization και μάλιστα με την ίδια ευκολία. Εξίσου εύκολο είναι να αλλάξει κανείς και τα ονόματα των συσκευών, στο μητρώο των Windows. Βέβαια, μετά από μία επανεκκίνηση τα ονόματα θα αποκτήσουν πάλι την αρχική τους τιμή, αλλά η "ζημία" μπορεί να έχει γίνει ήδη! Εξάλλου, ένας χρήστης με προχωρημένες γνώσεις, θα μπορούσε να επέμβει ακόμη και στον κώδικα ενός προγράμματος virtualization που αποτελεί προϊόν Ανοι-

χτού Λογισμικού. Με αυτόν τον τρόπο θα μπορούσε να δημιουργήσει ένα εικονικό περιβάλλον απαλλαγμένο πλήρως από τα ενοχοποιητικά στοιχεία που εξετάσαμε ως τώρα. Αναρωτιέστε τι μπορούμε να κάνουμε σε μια τέτοια περίπτωση; Μα, φυσικά, να στραφούμε την αγαπημένη μας Assembly. Ο compiler που συνοδεύει το περιβάλ-



Εικόνα 7
Σε ένα σύγχρονο σύστημα είναι απίθανο να υπάρχει μόνο μια κάρτα δικτύου. Συνεπώς, ο κώδικάς μας θα πρέπει να παίρνει τα στοιχεία για όλες τις κάρτες και να ελέγχει τη διεύθυνση MAC της καθεμίας ξεχωριστά.

λον C++ Builder επιτρέπει να γράφουμε inline Assembly πανεύκολα. Αυτό βέβαια δεν το αναφέρουμε σαν κάτι το πρωτοποριακό, αφού όλοι οι σύγχρονοι compilers επιτρέπουν κάτι τέτοιο. Όλα αυτά όμως είναι λίγο-πολύ γνωστά κι όσοι διαβάζουν το άρθρο περιμένουν να μάθουν πιο ενδιαφέροντα πράγματα. Έφτασε λοιπόν η ώρα να παρουσιάσουμε δύο προχωρημένες μεθόδους για την ανίχνευση ενός εικονικού περιβάλλοντος...



Εικόνα 8
Αν ο χρήστης αποφασίσει να επέμβει στα χαρακτηριστικά του VM, είναι πιθανό να αχρηστεύσει πολλούς από τους μηχανισμούς εντοπισμού των εικονικών μηχανών. Ευτυχώς για μάς, υπάρχουν και πιο αποτελεσματικές τεχνικές ανίχνευσης των εικονικών συστημάτων ;)

Μέθοδος "get version"

Πριν ξεκινήσουμε, να ξεκαθαρίσουμε ότι αυτός ο τρόπος ανίχνευσης λειτουργεί μόνο με τις μηχανές του VMware. Και για να προλάβουμε τα πιθανά (και δικαιολογημένα) παράπονα, έχετε υπόψη σας ότι η επόμενη μέθοδος που θα εξετάσουμε είναι πιο γενική — και δουλεύει παντού. Ας ξεκινήσουμε όμως με τη μέθοδο "get version" που ανέπτυξε ο Ken Kato.

Ως γνωστό, το VMware προσφέρει μερικά ειδικά εργαλεία (vmware-tools), τα οποία μπορούμε να προσθέσουμε στο guest OS ώστε να διευκολύνουμε τη συνεργασία του

με το host OS. Για την επικοινωνία αυτών των εργαλείων (που τρέχουν στην εικονική μηχανή) με τον έξω κόσμο (με το φυσικό σύστημα), χρησιμοποιείται ένα συγκεκριμένο IO port που ενσωματώνει το VMware σε όλες τις εικονικές μηχανές του. Αυτό το IO port αξιοποιεί και η μέθοδος του Ken Kato, ώστε να ζητήσει από το VMware την έκδοσή του! Ας ρίξουμε μια ματιά στον κώδικα:

```
__asm
{
        mov eax, 'VMXh' // the magic number
        mov ebx, 0;
        mov ecx, 0xA
        mov edx, 'VX'   // port number
        in eax, dx;     // read port
        cmp ebx, 'VMXh' // is it a reply from VMWare?
        jne notVmware
        jmp isVmware
notVmware:
        mov VMr, 0
        jmp done
isVmware:
        mov VMr, eax     // on return EAX returns the version
done:
}
```

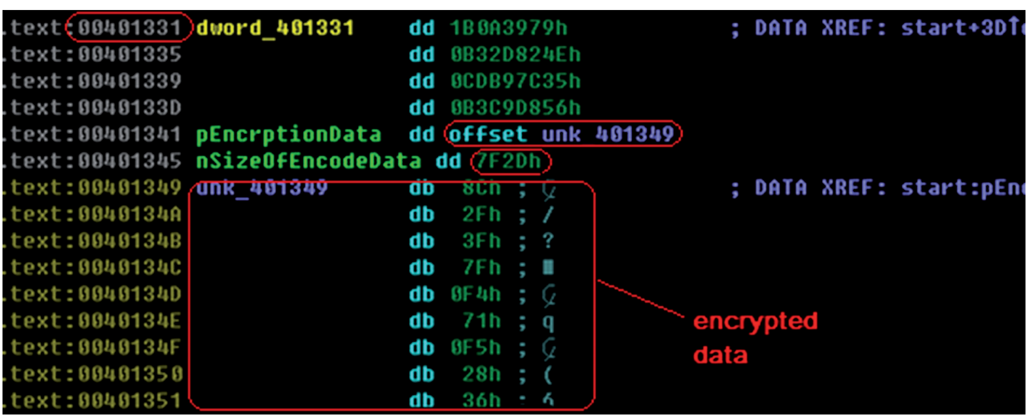
Αρχικά, φορτώνουμε στον καταχωρητή EAX την ειδική τιμή 0x564D5868 ("VMXh", σε μορφή ASCII). Αυτός ο ειδικός αριθμός (ονομάζεται και magic number) θα αποδοθεί αυτόματα στον καταχωρητή EBX, μετά την επιτυχημένη εκτέλεση της εντολής "πες μου την έκδοσή σου". Όπως αναφέραμε νωρίτερα, αυτή είναι η εντολή που θα στείλουμε στο VMware μέσα από το ειδικό IO port. Και, φυσικά, η έγκυρη μορφή της εντολής δεν είναι η πρόταση που βάλαμε στα εισαγωγικά, αλλά ο (δεκαεξαδικός) αριθμός 0xA. Γι' αυτούς τους λόγους, το πρόγραμμα προχωρά με τον καθαρισμό του καταχωρητή EBX (θα μπορούσαμε να του δώσουμε οποιαδήποτε τιμή εκτός από το magic number) και την απόδοση του 0xA στον καταχωρητή ECX (εκεί τοποθετούνται οι εντολές που στέλνουμε στο ειδικό IO port των μηχανών του VMware). Στη συνέχεια φορτώνουμε στον καταχωρητή EDX τον αριθμό που αντιστοιχεί στο ειδικό IO port. Πρόκειται για τον αριθμό 0x5658 ("VX", σε μορφή ASCII).

Μετά από τα παραπάνω, εκτελούμε μια εντολή IN. Σε αυτό το σημείο, όσοι έχουν ασχοληθεί με την Assembly είναι βέβαιο ότι αναρωτιούνται γιατί αποδώσαμε όλες αυτές τις τιμές στους καταχωρητές. Η απάντηση βρίσκεται στην εξής λεπτομέρεια: Το IO port που φυτεύει το VMware στις μηχανές του δεν συμπεριφέρεται όπως τα συνηθισμένα IO ports. Για την ακρίβεια, αποτελεί έναν πολύ διαφορετικό μηχανισμό που ενεργοποιείται όταν εκτελούμε μια εντολή IN κι *εφόσον* έχουμε δώσει κα-

τάλληλες τιμές σε κατάλληλους καταχωρητές. Με λίγα λόγια, ο εν λόγω μηχανισμός διαβάζει και γράφει σε περισσότερους καταχωρητές,από αυτούς που συμμετέχουν κανονικά σε μια εργασία IN ή OUT. Για αυτόν το λόγο, το συγκεκριμένο πρόγραμμα δεν θα μπορούσε να γραφτεί σε C. Οι βιβλιοθήκες της C που πραγματοποιούν την ανάγνωση και την εγγραφή σε διάφορα IO ports αδιαφορούν για τους καταχωρητές που χρησιμοποιεί το ειδικό IO port των μηχανών του VMware. Αν όλο αυτό σας μπέρδεψε, μπορείτε απλά να το ξεχάσετε. Στο κάτω-κάτω, σκοπός του άρθρου δεν είναι να γνωρίσουμε την αρχιτεκτονική x86, ούτε να μάθουμε Assembly.

Μετά την εκτέλεση της IN, το πρόγραμμα ελέγχει την τιμή του καταχωρητή EBX. Αν βρισκόμαστε μέσα σε εικονική μηχανή του VMware, ο EBX θα έχει αποκτήσει την ειδική τιμή (το "μαγικό αριθμό") που είχαμε δώσει αρχικά στον EAX. Αν όμως δεν βρισκόμαστε σε εικονική μηχανή του VMware, η τιμή του EBX δεν θα έχει επηρεαστεί καθόλου και θα είναι εκείνη που του δώσαμε εμείς (δηλαδή το μηδέν). Ο κώδικας, λοιπόν, συνεχίζει με αυτόν ακριβώς τον έλεγχο. Αν βρισκόμαστε σε εικονική μηχανή, η ρουτίνα ολοκληρώνεται αφού τοποθετήσουμε στη μεταβλητή VMr την έκδοση του VMware (τοποθετείται αυτόματα στον EAX, μετά την επιτυχημένη πρόσβαση στο IO port του VMware). Σε αντίθετη περίπτωση, η ρουτίνα ολοκληρώνεται με την μεταβλητή VMr να έχει την τιμή μηδέν.

Όπως αντιλαμβάνεστε, το πρόγραμμά μας καλεί τη ρουτίνα που μόλις παρουσιάσαμε και στη συνέχεια ελέγχει τη μεταβλητή VMr. Από την τιμή της μεταβλητής, το πρόγραμμα μπορεί να συμπεράνει με ασφάλεια το αν εκτελείται σε εικονική μηχανή του VMware ή όχι. Αν θέλετε να μάθετε περισσότερα για τις εντολές που αναγνωρίζει το ειδικό IO port των μηχανών του VMware, μπορείτε να μελετήσετε τις σημειώσεις του ίδιου του Ken Kato: `https://sites.google.com/site/chitchatvmback/backdoor`.



Εικόνα 9
Κρυπτογραφημένο τμήμα εκτελέσιμου! Ο disassembler αδυνατεί να ερμηνεύσει και να παρουσιάσει σωστά τον κώδικα…

Έλεγχος του IDTR

Ο κεντρικός επεξεργαστής ενός συστήματος προβλέπει μόνο ένα Interrupt Descriptor Table. Επιγραμματικά, να πούμε ότι ο εν λόγω πίνακας περιέχει τις διευθύνσεις των ρουτινών που καλούνται, όποτε προκύπτει κάποιο Interrupt. Πρόκειται δηλαδή για μία λίστα με τις διευθύνσεις των ISR (Interrupt Service Routine). Για την πρόσβαση σε αυτόν τον πίνακα χρησιμοποιείται ένας ειδικός καταχωρητής που ονομάζεται IDTR (Interrupt Descriptor Table Register). Εφόσον λοιπόν υπάρχει ένας μόνο IDT, η εικο-

νική μηχανή πρέπει να μεταφέρει το δικό της πίνακα σε κάποια ασφαλή τοποθεσία, ώστε να αποφευχθεί μια πιθανή διένεξη με τον αντίστοιχο πίνακα της φυσικής μηχανής. Ως αποτέλεσμα, η τιμή του IDTR σε μια εικονική μηχανή είναι διαφορετική από εκείνη του ίδιου register σε μια φυσική μηχανή. Ακριβώς αυτή τη διαφορά εντοπίζει ο κώδικας που έγραψε η Joanna Rutkowska (www.invisiblethingslab.com/itl/About.html) και με αυτόν τον τρόπο εκτιμά το αν εκτελείται μέσα σε εικονικό περιβάλλον ή όχι. Πιο συγκεκριμένα, αν ο IDTR παραπέμπει στη διεύθυνση 0x80ffffff, βρισκόμαστε σε φυσική μηχανή αρχιτεκτονικής x86. Αν ο IDTR παραπέμπει σε διεύθυνση της μορφής 0xe8XXXXXX, βρισκόμαστε σε εικονική μηχανή της Microsoft (Virtual PC). Αν πάλι παραπέμπει σε διεύθυνση της μορφής 0xffXXXXXX, βρισκόμαστε σε εικονική μηχανή του VMware. Ο κώδικας της Rutkowska ελέγχει τον προορισμό του IDTR κι από αυτόν εκτιμά το αν βρισκόμαστε σε εικονικό σύστημα ή σε φυσικό…

```
unsigned long get_idt_base (void) {
   unsigned char idtr[6];
   unsigned long idt = 0;
   _asm sidt idtr
   idt = *((unsigned long *)&idtr[2]);
   return (idt);
}
void __fastcall TForm1::Button6Click(TObject *Sender) {
   unsigned int idt_base = 0;
   idt_base = get_idt_base ();
   if ((idt_base >> 24) == 0xff) {
      MessageBox(NULL,L"VMwre Machine Detected!", L"Report", MB_
ICONINFORMATION);
      return;
   } else {
      MessageBox(NULL,L"Physical Machine Detected!", L"Report", MB_
ICONINFORMATION);
   }
}
```

Πάνω σε αυτό το θέμα έχουν γραφεί πραγματικά αριστουργήματα και εκτενείς έρευνες από την Rutkowska. Η ίδια, άλλωστε, πριν πολλά χρόνια είχε δημιουργήσει το περίφημο RedPill. Πρόκειται για ένα μικρό πρόγραμμα, το οποίο επιστρέφει την τιμή μηδέν όταν εκτελείται σε εικονική μηχανή ή μια μη μηδενική τιμή όταν εκτελείται σε φυσική μηχανή.

```
int swallow_redpill ()
{
   unsigned char m[2+4], rpill[] = "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
   *((unsigned*)&rpill[3]) = (unsigned)m;
   ((void(*)())&rpill)();
   return (m[5]>0xd0) ? 1 : 0;
}
```

Το εντυπωσιακό χαρακτηριστικό αυτού του προγράμματος είναι ότι καταφέρνει να διαβάσει την τιμή του IDTR, ενώ εκτελείται χωρίς προνόμια (non privileged mode). Αυτό βέβαια αποτελεί απαραίτητη προϋπόθεση για κάθε καλό exploit. Πάντως, μια έρευνα που έκανε ο Γερμανός Oliver Schneider το 2007 (http://blog.assarbad.net/wp-content/uploads/2007/04/redpill_getting_colorless.pdf), ότι η μέθοδος της Joanna Rutkowska δεν παρουσίαζε σταθερή συμπεριφορά.

Γενικότερα, η "επιστήμη" της ανίχνευσης των χαρακτηριστικών ενός εικονικού περιβάλλοντος δεν γνωρίζει όρια. Σκοπός του παρόντος άρθρου δεν είναι να καλύψει κάθε πιθανό σενάριο, αλλά να παρουσιάσει τον τρόπο σκέψης και εργασίας. Ελπίζουμε να διασκεδάσατε διαβάζοντάς το — και να διασκεδάσετε ακόμα περισσότερο εφαρμόζοντας στην πραξη τις τεχνικές που παρουσιάσαμε.

Anti-Disassembling και Anti-Debugging

Οι πρακτικές που περιγράψαμε παραπάνω δεν προστατεύουν την εφαρμογή μας από τη μη εξουσιοδοτημένη τροποποίηση. Αποτελούν μόνο ένα προπύργιο ασφαλείας, που δυσκολεύει την εργασία ενός αναλυτή (το disassembling και το debugging). Ένας αναλυτής, όμως, θα μπορούσε να καταργήσει τους ελέγχους που παρουσιάσαμε, πριν καν ξεκινήσει την εργασία του στο εικονικό περιβάλλον. Ακριβώς γι' αυτό, αν θέλουμε να αυξήσουμε την προστασία μιας εφαρμογής, πρέπει να καταφύγουμε και σε τεχνικές anti-debugging και anti-disassembling.

Οι τεχνικές anti-disassembling έχουν σαν σκοπό να μπερδέψουν τον disassembler και να δώσουν έναν πηγαίο κώδικα μηχανής (disassembled machine code) που είναι λανθασμένος και δεν βγάζει νόημα. Με τεχνικούς όρους, θα λέγαμε ότι αυτές οι μέθοδοι προστασίας αποτελούν άμυνα για τη στατική ανάλυση ενός εκτελέσιμου αρχείου. Το debugging όμως αποτελεί μια δυναμική τεχνική ανάλυσης, κατά την οποία ο κώδικας εκτελείται βήμα προς βήμα από κάποιον debugger (π.χ, από τον γνωστό μας Olly debugger). Οι τεχνικές anti-debugging προβλέπουν των ενσωμάτωση κατάλληλων τμημάτων κώδικα, κατά προτίμηση στην αρχή του προγράμματός μας. Μόλις αυτά τα τμήματα αντιληφθούν ότι ο χρήστης χρησιμοποιεί κάποιον debugger, τον οδηγούν σε endless loops ή, απλά, στην έξοδο του προγράμματος. Κι όλα αυτά ισχύουν στις καλές περιπτώσεις. Υπάρχουν περιπτώσεις που ο κώδικας anti-debugging προκαλεί BSOD on-demand ;)

Ένα πρόγραμμα που θέλουμε να είναι άρτιο ως προς την περιμετρική του ασφάλεια, θα πρέπει να ακολουθεί την εξής διάρθρωση:

1. Αρχή κώδικα
2. Τεχνικές VM detection
3. Τεχνικές anti-disassembling
4. Τεχνικές anti-debugging
5. Κώδικας προγράμματος

Στη συνέχεια, αφού μεταγλωττίσουμε τον πηγαίο κώδικα και πάρουμε το τελικό εκτελέσιμο, θα προχωρήσουμε στην κρυπτογράφησή του. Εργαλεία που πετυχαίνουν αυτό το τελικό βήμα υπάρχουν στο Internet και μάλιστα σε αφθονία. Για να κατανοήσετε πόσο δυσκολεύει η δουλειά ενός αναλυτή όταν ένα πρόγραμμα είναι κρυπτογραφημένο, ρίξτε μια ματιά στην εικόνα 9.

Για άλλη μια φορά, ελπίζουμε να απολαύσατε κι εσείς τη μικρή μας εισαγωγή στα χωράφια του Virtual Machine Detection. May the code be with you!

Skill: Intermediate
Tags: LAMP, PHP, MySQL, web, development

MySQL

by multiPetros

Web development σε περιβάλλον LAMP [Μέρος 7]

Στα δύο προηγούμενα άρθρα γνωρίσαμε αρκετές δυνατότητες της MySQL, αλλά η επαφή μας μαζί της περιοριζόταν στην κονσόλα. Έφτασε ο καιρός να αλλάξει αυτό: Σε αυτό το μέρος θα συνδυάσουμε τα υποσυστήματα του LAMP stack και θα δημιουργήσουμε εφαρμογές web που θα αλληλεπιδρούν με τη βάση προγραμματιστικά.

Μετά από όσα είδαμε στα δύο προηγούμενα άρθρα της σειράς, μπορούμε να αναπτύσσουμε βάσεις δεδομένων, να δημιουργούμε τους επιθυμητούς πίνακες και να προσθαφαιρούμε εγγραφές και πεδία. Επιπρόσθετα, μάθαμε να συντάσσουμε ερωτήματα προς τη βάση, ώστε να προσθαφαιρούμε περιεχόμενο ή απλά να το ανασύρουμε. Όλα αυτά, όμως, μάθαμε να τα κάνουμε χειροκίνητα κι από την κονσόλα. Για ν' αποκτήσουν πρακτική αξία οι γνώσεις που αποκτήσαμε, πρέπει να προχωρήσουμε στη διασύνδεση μεταξύ MySQL και PHP. Κάπως έτσι, θα εγκαταλείψουμε το αγαπημένο μας MySQL Monitor και θα αποκτήσουμε τον πλήρη έλεγχο των βάσεών μας, προγραμματιστικά. Η PHP διαθέτει ένα πλούσιο ρεπερτόριο συναρτήσεων γι' αυτή τη δουλειά.

Δικτυακή εφαρμογή με βάση δεδομένων

Προκειμένου να κατανοήσουμε καλύτερα όσα θα ακολουθήσουν, θα εξετάσουμε τον τρόπο λειτουργίας μιας δικτυακής εφαρμογής η οποία αξιοποιεί όλα τα υποσυστήματα του LAMP. Ακολουθούν τα γεγονότα που διαδραματίζονται στον server, όταν συνδέεται ένας χρήστης:

- Όλα ξεκινούν από τον web browser του χρήστη — τον συνήθη ταραχοποιό της ηρεμίας ενός server. Ο browser στέλνει ένα αίτημα HTTP και μ' αυτό ζητάει από τον server το περιεχόμενο μιας κάποιας σελίδας.

- Ο web server (δηλαδή ο Apache, εφόσον έχουμε LAMP stack), ξυπνάει από τον φαινομενικό του λήθαργο κι επεξεργάζεται το αίτημα. Διαπιστώνοντας ότι η ζητούμενη σελίδα αποτελεί ένα έγγραφο PHP κι ανακουφισμένος, αφού η επεξεργασία του δεν θα απασχολήσει τον ίδιο, διαβιβάζει το εν λόγω έγγραφο στον διερμηνέα της PHP.

- Ο δε διερμηνέας της PHP, εμφανώς ενοχλημένος αφού ποτέ δεν βρίσκει λίγη ησυχία ("Μα, όλοι έγγραφα PHP δημιουργούν πια; Κανείς δεν γράφει σκέτη HTML;"), ξεκινάει την εκτέλεση του κώδικα. Ξαφνικά, συναντά μια συνάρτηση σύνδεσης με τη βάση δεδομένων κι ένα ερώτημα (query). Η χαρά του δεν περιγράφεται! Συνδέεται με τη βάση, διαβιβάζει το ερώτημα κι απολαμβάνει μερικά millisecond ανάπαυσης...

- Ο MySQL server ελέγχει σχολαστικά τα στοιχεία της σύνδεσης κι εφόσον είναι έγκυρα, ξεκινά την εκτέλεση του ερωτήματος. Τα δεδομένα που προκύπτουν αποστέλλονται στο διερμηνέα της PHP, ώστε να φύγουν όσο το δυνατόν γρηγορότερα από την ευθύνη του MySQL server ;)

- Ο διερμηνέας της PHP σηκώνει για μια ακόμη φορά τα μανίκια και συνεχίζει την εκτέλεση του κώδικα, αξιοποιώντας και τα δεδομένα που έλαβε από τη βάση. Όταν ολοκληρωθεί η εκτέλεση του PHP script, στέλνει το τελικό προϊόν στον Apache, με την ελπίδα ότι αυτός θα τον αφήσει ήσυχο — έστω και για λίγο!

- Ο Apache λαμβάνει όσα προέκυψαν από την εκτέλεση του κώδικα PHP και τα σερβίρει στον web browser του χρήστη...

Διασύνδεση εξαρτημάτων

Τώρα που έχουμε τη γενική εικόνα για το πώς συνεργάζονται τα εξαρτήματα του LAMP, μπορούμε να προχωρήσουμε στην υλοποίηση ενός τέτοιου συστήματος. Ουσιαστικά, αυτό που θέλουμε να πετύχουμε είναι η σύνδεση ενός προγράμματος σε PHP με τη MySQL. Για αυτή την εργασία θα χρησιμοποιήσουμε τις συναρτήσεις της βιβλιοθήκης mysqli.

Καταρχάς, για να αποκτήσουμε πρόσβαση σε μία βάση δεδομένων της MySQL πρέπει πρώτα να πραγματοποιήσουμε μια σύνδεση προς τον MySQL server. Αυτό επιτυγχάνεται με τη χρήση της συνάρτησης mysqli_connect ή με τη δημιουργία ενός αντικειμένου mysqli. Σε κάθε περίπτωση, πρέπει να καθορίσουμε τον server, το όνομα του χρήστη, τον αντίστοιχο κωδικό πρόσβασης, καθώς και το όνομα της επιθυμητής βάσης δεδομένων. Ας δούμε τώρα πώς θα μπορούσαμε να συνδεθούμε στη βάση δεδομένων gbook, που δημιουργήσαμε στα προηγούμενα άρθρα της σειράς. Για τη σύνδεση θα χρησιμοποιήσουμε το λογαριασμό χρήστη gbook_web και θα δώσουμε ένα παράδειγμα και για τους δύο τρόπους σύνδεσης.

Σύνδεση με τη συνάρτηση mysqli_connect:

```
$db_con = mysqli_connect('localhost', 'gbook_web', 'securepass', 'gbook') ;
```

Σύνδεση με τη δημιουργία αντικειμένου mysqli:

```
$db_obj = new mysqli('localhost', 'gbook_web', 'securepass', 'gbook') ;
```

Αμέσως μετά, επιβάλλεται να πραγματοποιήσουμε έναν έλεγχο για το αν ήταν επιτυχημένη η σύνδεση. Ο έλεγχος αυτός πραγματοποιείται πανεύκολα, με τη βοήθεια της συνάρτησης mysqli_connect_errno. Όταν η σύνδεση έχει αποτύχει, η εν λόγω συνάρτηση επιστρέφει τον αντίστοιχο κωδικό σφάλματος. Διαφορετικά, η ίδια συνάρτηση επιστρέφει τον αριθμό μηδέν. Επομένως, αρκεί να καλέσουμε τη συνάρτηση και να ελέγξουμε το αποτέλεσμα.

Εκτέλεση ερωτημάτων

Εφόσον η σύνδεση ήταν επιτυχής, μπορούμε να προχωρήσουμε στην αποστολή ερωτημάτων προς τη βάση. Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε τη συνάρτηση mysqli_query(connection_name, query_string). Εναλλακτικά, αν έχουμε ακολουθήσει την αντικειμενοστραφή προσέγγιση κι έχουμε δημιουργήσει ένα αντικείμενο mysqli, μπορούμε να χρησιμοποιήσουμε τη μέθοδο query(query_string). Ακολουθεί ένα παράδειγμα για κάθε μέθοδο.

Διαδικαστική προσέγγιση:

```
$result_res = mysqli_query($db_con, 'select name, msg, date from users,
messages where users.id = messages.user') ;
```

Αντικειμενοστραφής προσέγγιση:

```
$result_obj = db_obj->query('select name, msg, date from users, messages
where users.id = messages.user') ;
```

Παρατηρείστε ότι αν και στο MySQL Monitor ήταν απαραίτητο να τερματίζουμε κάθε πρόταση με το χαρακτήρα του ελληνικού ερωτηματικού, όταν χρησιμοποιούμε τη βιβλιοθήκη mysqli δεν χρειάζεται να "κλείνουμε", με κάποιον τρόπο, τα ερωτήματα. Τα ελληνικά ερωτηματικά που φαίνονται στα παραδείγματα αποτελούν συντακτικό

στοιχείο της PHP και τερματίζουν τις αντίστοιχες εντολές. Στην πρώτη περίπτωση επιστρέφεται μια μεταβλητή τύπου resource (στο παράδειγμα πρόκειται για την $result_res), ενώ στη δεύτερη ένα αντικείμενο ($result_obj). Από αυτή τη μεταβλητή ή το αντικείμενο μπορούμε να αντλήσουμε τα δεδομένα που προέκυψαν από την εκτέλεση του ερωτήματος στη βάση. Αν κατά την εκτέλεση του ερωτήματος προκύψει κάποιο πρόβλημα, αντί της μεταβλητής τύπου resource ή του αντικειμένου, θα λάβουμε την τιμή false. Έτσι, είναι πολύ εύκολο να τσεκάρουμε αν το ερώτημα εκτελέστηκε επιτυχώς ή όχι.

Με την ευκαιρία, αξίζει να υπογραμμίσουμε μια λεπτομέρεια. Σε πολλές δικτυακές εφαρμογές, τα ερωτήματα προς τη βάση σχηματίζονται σύμφωνα με δεδομένα που εισάγει ο χρήστης. Έτσι, αν ο κώδικάς μας δεν είναι προσεκτικά γραμμένος, ενδέχεται να είναι ευπαθής σε επιθέσεις SQL injection. Για να αποφύγουμε αυτό το ενδεχόμενο, πρέπει κάθε φορά που σχηματίζεται ένα ερώτημα και *πριν* σταλεί προς τη βάση, να περάσει από κάποιους ελέγχους. Επομένως, από την άποψη της δομής του προγράμματός μας, είναι προτιμότερο να σχηματίζουμε το εκάστοτε ερώτημα σε μια μεταβλητή string, παρά να το "φυτεύουμε" απευθείας μέσα στη συνάρτηση mysqli_query ή στη μέθοδο query. Με αυτόν τον τρόπο θα έχουμε τη δυνατότητα να πραγματοποιήσουμε όλους τους απαραίτητους ελέγχους, πριν εκτελεστεί το εκάστοτε ερώτημα. Όπως καταλαβαίνετε, αυτή η πρακτική δεν αντιμετωπίζει το πρόβλημα των επιθέσεων, αλλά καθιστά τον κώδικα πιο καθαρό και μας επιτρέπει να λάβουμε όλα τα απαραίτητα μέτρα.

Ανάκτηση αποτελεσμάτων

Για την ανάκτηση των αποτελεσμάτων από την εκτέλεση ενός ερωτήματος, προσφέρονται διάφορες συναρτήσεις. Σε καθεμία από αυτές, δίνουμε σαν παράμετρο τη μεταβλητή τύπου resource, με τα αποτελέσματα. Πριν προχωρήσουμε όμως, πρέπει να πούμε δυο λόγια για τη φύση των αποτελεσμάτων. Εφόσον τα ερωτήματα προς τη βάση εκτελούνται πάνω σε κάποιον πίνακα, τα αποτελέσματα είναι κι αυτά ένας πίνακας! Αυτός ο πίνακας ενδέχεται να έχει μια γραμμή και πολλές στήλες, μια στήλη και πολλές γραμμές, πολλές γραμμές και πολλές στήλες ή μια γραμμή και μία στήλη. Αυτή η διευκρίνιση μπορεί να μοιάζει περιττή, αλλά πρέπει να έχουμε ξεκάθαρο στο μυαλό μας ότι τα αποτελέσματα έρχονται πάντα με τη μορφή ενός πίνακα. Ας δούμε τώρα τι προσφέρει καθεμία από τις συναρτήσεις ανάκτησης των αποτελεσμάτων.

- **mysqli_fetch_row():** Επιστρέφει τα δεδομένα μιας γραμμής από τον πίνακα των αποτελεσμάτων και αυξάνει το δείκτη της τρέχουσας γραμμής. Έτσι, την επόμενη φορά που θα εκτελέσουμε τη συνάρτηση, θα επιστρέψει τα δεδομένα της επόμενης γραμμής κ.ο.κ. Αν η γραμμή που λάβαμε απαρτίζεται από πολλές στήλες, τα δεδομένα κάθε στήλης τοποθετούνται σε ξεχωριστές θέσεις ενός "αριθμητικού πίνακα" (numeric array). Σε αυτούς τους πίνακες της PHP, για την αναφορά σε κάποιο στοιχείο χρησιμοποιούνται αριθμητικοί δείκτες (γενική μορφή: my_array[index_number]).

- **mysqli_fetch_assoc():** Λειτουργεί όπως η προηγούμενη. Ωστόσο, αν η γραμμή που πήραμε απαρτίζεται από πολλές στήλες, τα δεδομένα των στηλών τοποθετούνται στις θέσεις ενός "συσχετιζόμενου πίνακα" (associative array). Σε αυτούς τους πίνακες της PHP, για την αναφορά σε κάποιο στοιχείο χρησιμοποιούνται λέξεις-κλειδιά (γενική μορφή: my_array['element_name']).

- **mysqli_fetch_array():** Συμπεριφέρεται όπως οι προηγούμενες και συνδυά-

ζει τη λειτουργικότητα και των δύο. Αν η γραμμή των αποτελεσμάτων που πήραμε απαρτίζεται από πολλές στήλες, τα δεδομένα των στηλών τοποθετούνται σε έναν αριθμητικό πίνακα, σε έναν συσχετιζόμενο ή και στους δύο.

- **mysqli_fetch_object():** Όπως κι όλες οι υπόλοιπες, επιστρέφει κάθε φορά μια γραμμή των αποτελεσμάτων. Ωστόσο, τα σχετικά δεδομένα δεν τοποθετούνται σε κάποιον πίνακα, αλλά σε ένα αντικείμενο! Τα στοιχεία, από κάθε στήλη της γραμμής των αποτελεσμάτων, αποθηκεύονται σε μια ομώνυμη ιδιότητα του αντικειμένου.

Για την ώρα, ενδέχεται να μην έχετε κατανοήσει πλήρως το πώς επιστρέφονται τα δεδομένα, ούτε τις διαφορές ανάμεσα στα είδη πινάκων που αναφέραμε. Ωστόσο, όταν δούμε μερικά παραδείγματα, το τοπίο θα ξεκαθαρίσει αμέσως.

Δεν ξέρουμε αν το προσέξατε, αλλά όλα τα παραπάνω ισχύουν αν έχουμε ακολουθήσει τη "διαδικαστική" προσέγγιση στην εκτέλεση ενός ερωτήματος. Αν έχουμε ακολουθήσει την αντικειμενοστραφή προσέγγιση της βιβλιοθήκης mysqli, τα πράγματα είναι εξίσου απλά. Για καθεμία από τις συναρτήσεις που αναφέραμε προηγουμένως, προσφέρεται και μια αντίστοιχη μέθοδος, με το ίδιο όνομα (fetch_row, fetch_assoc, fetch_array και fetch_object).

Πλήρης ανάκτηση

Κάθε φορά που εκτελούμε ένα ερώτημα προς τη βάση, είναι σίγουρο ότι θέλουμε να εξετάσουμε *όλα* τα σχετικά αποτελέσματα. Έτσι, ανεξάρτητα από ποια συνάρτηση ή μέθοδο θα επιστρατεύσουμε για την ανάκτηση των αποτελεσμάτων, είναι βέβαιο ότι θα θελήσουμε να σχηματίσουμε έναν βρόχο. Με τη βοήθεια του βρόχου θα διαβάσουμε μία προς μία όλες τις γραμμές των αποτελεσμάτων και θα κάνουμε τις απαιτούμενες ενέργειες. Νομίζουμε ότι όλο αυτό είναι απλό και αυτονόητο. Όμως, για να σχηματίσουμε τον απαιτούμενο βρόχο πρέπει να ξέρουμε πόσες φορές θα επαναληφθεί ο βρόχος, άρα να ξέρουμε πόσες γραμμές αποτελεσμάτων έχουμε. Για το σκοπό αυτό προσφέρεται η συνάρτηση mysqli_num_rows(). Σε αυτή δίνουμε τη μεταβλητή τύπου resource με τα αποτελέσματα κι αυτή μας επιστρέφει το πλήθος των γραμμών των αποτελεσμάτων. Αν έχουμε ακολουθήσει την αντικειμενοστραφή προσέγγιση, μπορούμε να χρησιμοποιήσουμε την ιδιότητα num_rows του αντικειμένου με τα αποτελέσματα. Ας δούμε ένα παράδειγμα "εκτύπωσης" των αποτελεσμάτων, μετά την εκτέλεση του ερωτήματος που είδαμε νωρίτερα...

```
for ($i=0; $i < results_obj->num_rows; $i++) {
$message = results_obj->fetch_assoc() ;
echo '<pre>-- message no '.$i.' --<br>' ;
echo 'username: '.$message['name'].' <br>' ;
echo 'message : '.$message['msg'].' <br>' ;
echo 'date    : '.$message['date'].' <br>' ;
echo '----------------------------</pre>' ;
}
```

Για το πλήθος των επαναλήψεων του βρόχου χρησιμοποιήσαμε την ιδιότητα num_rows, του αντικειμένου με τα αποτελέσματα ($results_obj). Στη συνέχεια, για την ανάκτηση κάθε γραμμής των αποτελεσμάτων, χρησιμοποιήσαμε τη μέθοδο

fetch_assoc(), η οποία είναι αντίστοιχη της συνάρτησης mysqli_fetch_ assoc(). Έτσι, κάθε γραμμή των αποτελεσμάτων τοποθετείται σε έναν πίνακα τύπου associative. Το όνομα που επιλέξαμε για τον πίνακα ήταν το $message κι έτσι, για να πάρουμε τα μεμονωμένα στοιχεία κάθε στήλης, χρησιμοποιήσαμε το όνομα της στήλης σαν δείκτη για τον πίνακα. Παρακάτω φαίνεται ένας παρόμοιος βρόχος, μόνο που στη θέση της μεθόδου fetch_assoc() χρησιμοποιούμε τη μέθοδο fetch_object. Ρίξτε μια ματιά και είμαστε σίγουροι ότι θα κατανοήσετε τα πάντα αμέσως...

```
for($i=0; $i < results_obj->num_rows; $i++){
$message = results_obj->fetch_object() ;
echo '<pre>-- message no '.$i.' --<br>' ;
echo 'username: '.$message->name.' <br>' ;
echo 'message : '.$message->msg.' <br>' ;
echo 'date    : '.$message->date.' <br>' ;
echo '---------------------------</pre>' ;
}
```

Άλλου τύπου αποτελέσματα

Μέχρι στιγμής εξετάσαμε την ανάκτηση αποτελεσμάτων από την εκτέλεση ενός ερωτήματος επιλογής (αναζήτησης). Τι γίνεται όταν εκτελούμε ερωτήματα εισαγωγής, ενημέρωσης ή διαγραφής; Σε αυτές τις περιπτώσεις, το μόνο στοιχείο που θα μπορούσαμε να θεωρήσουμε σαν αποτέλεσμα είναι το πλήθος των γραμμών που τροποποιήθηκαν. Κατά συνέπεια, όταν εκτελούμε ένα ερώτημα εισαγωγής, ενημέρωσης ή διαγραφής, το αποτέλεσμα που λαμβάνουμε δεν είναι κάποια μεταβλητή τύπου resource, ούτε κάποιο αντικείμενο. Αν το ερώτημα εκτελεστεί σωστά, λαμβάνουμε ένα true και στην αντίθετη περίπτωση ένα false. Αν θέλουμε να μάθουμε πόσες γραμμές επηρεάστηκαν από την εκτέλεση του ερωτήματος, αρκεί να χρησιμοποιήσουμε τη συνάρτηση mysqli_affected_rows(). Σε αυτή τη συνάρτηση δίνουμε ως παράμετρο τη μεταβλητή τύπου resource, στην οποία έχουμε αντιστοιχίσει τη σύνδεση στη βάση δεδομένων. Με βάση το δικό μας παράδειγμα (το πρώτο που συναντήσαμε στο άρθρο), πρόκειται για τη μεταβλητή $db_con. Την ίδια συμπεριφορά επιδεικνύει και η μέθοδος affected_rows(). Ας δούμε από ένα παράδειγμα.

Με χρήση της συνάρτησης:

```
echo 'There are '.mysqli_affected_rows($db_con).' lines affected';
```

Με χρήση της μεθόδου:

```
echo 'There are '.db_obj->affected_rows().' lines affected';
```

Προκατασκευασμένες προτάσεις

Μια διαφορετική προσέγγιση στην επικοινωνία με τη βάση δεδομένων συνιστά η χρήση προκατασκευασμένων προτάσεων. Οι προτάσεις αυτές επιτρέπουν την προετοιμασία της MySQL για την εκτέλεση ενός ερωτήματος, μέσω της αποστολής της γενικής δομής του ερωτήματος. Αφού "προειδοποιήσουμε" τη βάση δεδομένων για τη μορφή του ερωτήματος, μπορούμε να στείλουμε και τις λεπτομέρειες: Τα δεδομένα που θα καταστήσουν το ερώτημά μας συγκεκριμένο.

Η χρήση των προκατασκευασμένων προτάσεων είναι ιδιαίτερα βολική κι αποδοτική, όταν επιθυμούμε τη μαζική εισαγωγή δεδομένων στη βάση. Επιπρόσθετα, η ίδια μέθοδος προσφέρει κι ένα ακόμα επίπεδο προστασίας, ενάντια στις επιθέσεις SQL Injection. Στη συνέχεια θα δούμε ένα παράδειγμα εισαγωγής δεδομένων με τη βοήθεια ενός προκατασκευασμένου ερωτήματος:

```
0    $db_obj = new mysqli('localhost', 'gbook_web', 'securepass', 'gbook');
1    $query = "select msg from messages, users where
     users.id = messages.user and users.name = ?";
2    $stmt = $db_obj->prepare($query) ;
3    $stmt->bind_param('lucky reader') ;
4    $stmt->bind_result($msg) ;
5    $stmt->execute() ;
6    for($i=0; $i < $stmt->num_rows; $i++){
7            $message = $ stmt->fetch() ;
8            echo '<pre>-- message no '.$i.' --<br>' ;
9            echo 'message : '.$msg.' <br>' ;
10           echo '----------------------------</pre>' ;
11   }
```

Στη γραμμή 0 πραγματοποιούμε μια σύνδεση προς τη βάση και δημιουργούμε ένα αντίστοιχο αντικείμενο ($db_obj). Στη γραμμή 1 σχηματίζουμε τη γενική μορφή των ερωτημάτων μας και την αποθηκεύουμε σε μια μεταβλητή. Το μόνο αξιοπρόσεκτο στοιχείο εδώ, αποτελεί το αγγλικό ερωτηματικό. Στη θέση του αγγλικού ερωτηματικού θα τοποθετούνται αργότερα τα δεδομένα, που θα στέλνουμε κάθε φορά για τη συμπλήρωση του ερωτήματος. Στη γραμμή 2 χρησιμοποιούμε το αντικείμενο $db_obj και, ουσιαστικά, δηλώνουμε στη βάση τη γενική μορφή των ερωτημάτων που θα εκτελούμε. Για το σκοπό αυτό επιστρατεύουμε τη μέθοδο prepare(), στην οποία δίνουμε ως παράμετρο —τι άλλο;— τη μεταβλητή με τη γενική μορφή του ερωτήματος. Η μέθοδος επιστρέφει ένα αντικείμενο (το ονομάσαμε $stmt), μέσω του οποίου μπορούμε να χειριζόμαστε πλέον το προκατασκευασμένο ερώτημα.

Στη γραμμή 3 δίνουμε τα δεδομένα που θέλουμε να τοποθετηθούν στο γενικό ερώτημα, ώστε να αποκτήσει συγκεκριμένη μορφή. Αυτό επιτυγχάνεται με τη μέθοδο bind_param(). Σημειώστε ότι η γενική μορφή του ερωτήματος θα μπορούσε να περιέχει πολλά κενά προς συμπλήρωση (πολλά αγγλικά ερωτηματικά). Σε αυτή την περίπτωση, θα έπρεπε να δώσουμε μια τιμή για κάθε κενό.

Στη γραμμή 4, χρησιμοποιώντας τη μέθοδο bind_result, δηλώνουμε ότι κάθε φορά που θα εκτελείται το ερώτημα θέλουμε τα αποτελέσματα να τοποθετούνται στη μεταβλητή $msg. Σημειώστε ότι αν τα αποτελέσματα (οι γραμμές) που προκύπτουν από την εκτέλεση του ερωτήματος περιλαμβάνουν πολλές στήλες, θα πρέπει να δηλώσουμε μια μεταβλητή για κάθε στήλη. Το δικό μας ερώτημα, πάντως, θα επιστρέφει αποτελέσματα (γραμμές) με μία μόνο στήλη (με ένα στοιχείο).

Στη γραμμή 5 εκτελείται (επιτέλους) το ερώτημα! Ακολουθεί ένα βρόχος, μέσα στον οποίο λαμβάνουμε και τυπώνουμε τα αποτελέσματα. Κάθε φορά που εκτελείται ο βρόχος, εκτελούμε τη μέθοδο fetch. Αυτή φροντίζει να ενημερώσει κατάλληλα τις

μεταβλητές που έχουμε ορίσει με τη μέθοδο bind_result — στην περίπτωσή μας πρόκειται μόνο για τη μεταβλητή $msg.

Αποσύνδεση

Κάθε φορά που ολοκληρώνουμε την επεξεργασία των δεδομένων που πήραμε από τη βάση, είναι φρόνιμο να αποδεσμεύουμε τους σχετικούς πόρους. Αυτό γίνεται με τη συνάρτηση mysqli_free_result(), στην οποία δίνουμε ως παράμετρο τη μεταβλητή τύπου resource, που διατηρεί τα αποτελέσματα. Αν έχουμε ακολουθήσει την αντικειμενοστραφή οδό, αρκεί να χρησιμοποιήσουμε τη μέθοδο free() επί του αντικειμένου με τα αποτελέσματα. Επίσης, κάθε φορά που ολοκληρώνουμε την εργασία μας με κάποια βάση, είναι καλό να πραγματοποιούμε και μια αποσύνδεση. Αυτό γίνεται με τη συνάρτηση mysqli_close() ή τη μέθοδο close(). Ακολουθούν τα σχετικά παραδείγματα.

Με συναρτήσεις:

```
mysqli_free_result($result_res) ;
mysqli_close($db_con);
Με αντικείμενα:
$result_obj->free() ;
$db_obj->close() ;
```

Αυτόματη απόδραση!

Όπως έχουμε αναφέρει και σε προηγούμενα άρθρα της σειράς, δεν πρέπει να ξεχνάμε ποτέ το φιλτράρισμα των δεδομένων που εισάγει ο χρήστης! Στα παραδείγματά μας, για λόγους απλότητας και μόνο, παραλείψαμε εντελώς αυτό το βήμα. Κάτι τέτοιο δεν πρέπει να συμβαίνει επ' ουδενί λόγω σε μια πραγματική δικτυακή εφαρμογή. Πέραν από τις συναρτήσεις φιλτραρίσματος που έχουμε συναντήσει ως τώρα, για την εισαγωγή δεδομένων μέσα σε βάσεις δεδομένων πρέπει να γίνεται χρήση της addslashes(). Όπως έχουμε πει, η MySQL ερμηνεύει ορισμένους χαρακτήρες (τους λεγόμενους και "ειδικούς") με ιδιαίτερο τρόπο. Για να εξασφαλίσουμε ότι ένας τέτοιος χαρακτήρας θα αντιμετωπιστεί σαν όλους τους άλλους, πρέπει να τοποθετήσουμε μπροστά του ένα slash. Αυτός ο χαρακτήρας ονομάζεται και χαρακτήρας διαφυγής, ενώ η χρήση του συνιστά αυτό που ονομάζουμε character escaping. Η συνάρτηση addslashes, λοιπόν, τοποθετεί αυτόματα το χαρακτήρα διαφυγής, πριν από κάθε χαρακτήρα που επιδέχεται ειδικής ερμηνείας. Ακριβώς γι' αυτό, είναι φρόνιμο να την εφαρμόζουμε σε όλα τα δεδομένα που δίνει ο χρήστης και τα οποία προορίζονται για τη βάση. Η συνάρτηση επιστρέφει μια μεταβλητή string, με τα δεδομένα που της δώσαμε και τους απαραίτητους χαρακτήρες διαφυγής *πριν* από κάθε ειδικό χαρακτήρα.

GuestBook με τη MySQL

Πλέον, διαθέτουμε όλα τα απαραίτητα εφόδια για να αναβαθμίσουμε την εφαρμογή GuestBook. Κάνοντας λόγο για αναβάθμιση, εννοούμε την προσθήκη της ικανότητας διαχείρισης μιας βάσης δεδομένων για την αποθήκευση και την ανάκτηση των μηνυμάτων που αφήνουν οι χρήστες. Στη συνέχεια παραθέτουμε τον ανανεωμένο κώδικα. Για λόγους οικονομίας χώρου, έχουμε παραλείψει τα τμήματα του προγράμματος

που παραμένουν αναλλοίωτα. Εσείς, όμως, μπορείτε να κατεβάσετε ολόκληρο τον πηγαίο κώδικα από το http://bit.ly/dh030msqlgb. Η αρίθμηση που χρησιμοποιούμε λαμβάνει υπόψη και τις γραμμές που παραλείψαμε. Αυτό το κάναμε για να μπορείτε να μελετήσετε την ανάλυση που ακολουθεί, ακόμα κι αν έχετε μπροστά σας το αρχείο με ολόκληρο τον κώδικα.

```
008   $saved_good = true;
009   $empty_form = false;
010
011   $db = new mysqli('localhost','gbook_web','securepass','gbook');
012   $id = null;
013
014   if(isset($_POST['nickname']) && isset($_POST['message'])) {
015         $nickname = addslashes(trim(htmlspecialchars ($_POST['nickname'])));
016         $message = addslashes(trim(htmlspecialchars ($_POST['message'])));
017         if($nickname == "" || $message == "") {
018                 $empty_form = true;
019         } else {
020                 $uid_result = $db->query("select id from users where name = '$nickname'");
021                 if($uid_result !== false && $uid_result->num_rows > 0) {
022                         $id = $uid_result->fetch_row()[0];
023                         $add_msg = $db->query("insert into messages values (NULL, '$id', '$message', now())");
024                 } else {
025                         $add_usr = $db->query("insert into users values (NULL, '$nickname')");
026                         $uid_result = $db->query("select id from users where name = '$nickname'");
027                         $id = $uid_result->fetch_row()[0];
028                         $add_msg = $db->query("insert into messages values (NULL, '$id', '$message', now())");
029                 }
030                 $saved_good = ($add_msg !== false) ? true : false;
031                 $uid_result->free();
032         }
033   }
034   $entries = $db->query("select name, msg, date from messages, users where messages.users = users.id order by date desc");
```

```
035   ?>
[…]
086         if($entries->num_rows < 1) {
087               echo "<li><h3 class=\"ui-li-heading\">Oops!</h3>" ;
088               echo "<p>No messages for the moment... </p>" ;
089         } else {
090               for($i=0; $i<$entries->num_rows; $i++) {
091                     $entry = $entries->fetch_object() ;
092                     echo "<li><p class=\"ui-li-aside ui-li-desc\"><em>@ ".stripslashes($entry->date)." </em></p>" ;
093                     echo "<h3 class=\"ui-li-heading\"> ".stripslashes($entry->name)." </h3>" ;
094                     echo "<p class=\"ui-li-desc\"><strong>Message:</strong> ".stripslashes($entry->msg)." </p></li>" ;
095               }
096         }
097         $entries->free() ;
098         $db->close() ;
```

Εισαγωγή μηνυμάτων

Με μια γρήγορη ματιά στον τροποποιημένο κώδικα, διαπιστώνουμε εύκολα ότι η λογική του προγράμματος παραμένει ίδια. Διαφοροποίηση παρατηρείται μόνο στον τρόπο αποθήκευσης και ανάκτησης των προς εμφάνιση μηνυμάτων. Στη γραμμή 11 πραγματοποιούμε μια σύνδεση στη βάση, δημιουργώντας ένα αντικείμενο mysqli. Στη συνέχεια, έως και τη γραμμή 19, εκτελούμε το γνωστό έλεγχο για τη συμπλήρωση της φόρμας από το χρήστη. Βέβαια, στις γραμμές 15 και 16, όπου γίνεται το λεγόμενο sanitization, χρησιμοποιούμε πλέον και τη συνάρτηση addslashes. Στη γραμμή 20 εκτελούμε ένα ερώτημα επιλογής, με το οποίο θα είμαστε σε θέση να ελέγξουμε (γραμμή 21) αν το όνομα χρήστη υπάρχει ήδη στον πίνακα users της βάσης. Αν υπάρχει, λαμβάνουμε το id του (το πρώτο στοιχείο/πεδίο του αποτελέσματος). Στη συνέχεια, στη γραμμή 23 εκτελούμε ένα ερώτημα εισαγωγής δεδομένων στον πίνακα messages της βάσης.

Στην περίπτωση που δεν υπάρχει χρήστης με το συγκεκριμένο όνομα μεταβαίνουμε στη γραμμή 25, όπου και εκτελούμε ένα ερώτημα εισαγωγής. Εδώ προσθέτουμε τον (προφανώς) νέο χρήστη, στον πίνακα users. Αμέσως μετά εκτελούμε ένα ερώτημα επιλογής, ώστε να μάθουμε το id που αποδόθηκε αυτόματα στο χρήστη (όπως θα θυμόσαστε από προηγούμενο άρθρο της σειράς, είχαμε ορίσει την αυτόματη συμπλήρωση του πεδίου id). Μετά, εκτελούμε ένα ερώτημα εισαγωγής στον πίνακα messages, για να προσθέσουμε το μήνυμα του χρήστη (γραμμή 28).

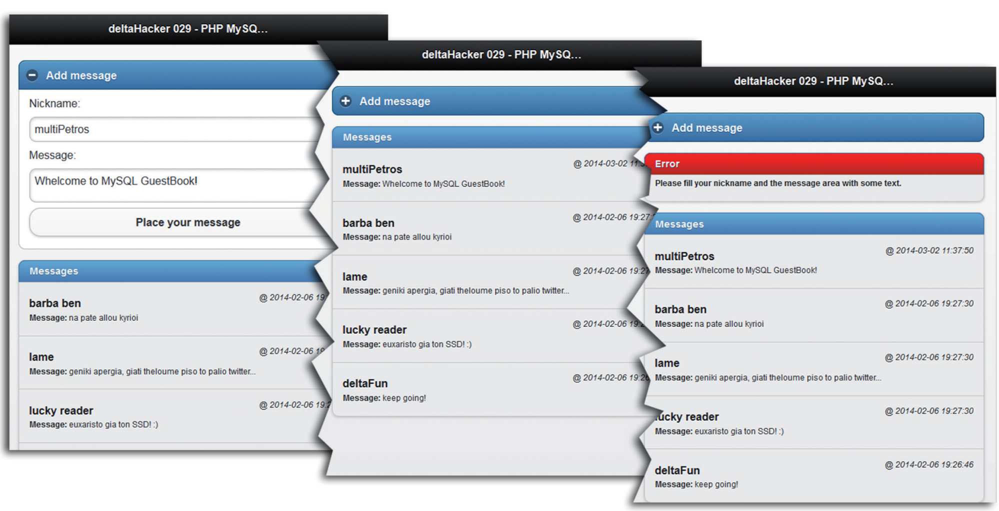
Κάθε φορά που προσθέτουμε ένα μήνυμα στον πίνακα messages, αποθηκεύουμε το αποτέλεσμα από την εκτέλεση του σχετικού ερωτήματος στη μεταβλητή $add_msg. Στη γραμμή 30 ελέγχουμε την τιμή της συγκεκριμένης μεταβλητής, για να διαπιστώσουμε αν πήγαν όλα καλά, ή όχι. Στη γραμμή 31 αποδεσμεύουμε/καταστρέφουμε το αντικείμενο $uid_result, που δημιουργήθηκε όταν εκτελέσαμε το ερώτημα εύρεσης

του χρήστη. Για αυτή τη δουλειά καλούμε τη μέθοδο free(). Μήπως αναρωτιέστε γιατί δεν κάνουμε το ίδιο και για το αντικείμενο $add_msg; Αυτό το αντικείμενο προέκυψε από ένα ερώτημα τροποποίησης της βάσης και ως εκ τούτου, δεν περιλαμβάνει κάποιου είδους αποτελέσματα και δεν δεσμεύει ιδιαίτερο χώρο.

Ανάκτηση μηνυμάτων

Μόλις εξετάσαμε το τμήμα του προγράμματος που αποθηκεύει τα μηνύματα και τους νέους χρήστες. Απλό δεν ήταν; Ας προχωρήσουμε τώρα στο τμήμα του προγράμματος που αναλαμβάνει την ανάκτηση των μηνυμάτων. Αυτό ξεκινά στη γραμμή 34, όπου και εκτελούμε ένα ερώτημα επιλογής (αναζήτησης). Με αυτό το ερώτημα ζητάμε δεδομένα από δύο πίνακες (messages και users) και, ουσιαστικά, αναζητάμε τα μηνύματα που έχουν αφήσει οι διάφοροι χρήστες. Ο ανανεωμένος κώδικας συνεχίζεται στη γραμμή 86, όπου ελέγχουμε το αν επιστράφηκαν δεδομένα (δηλαδή το αν υπάρχουν μηνύματα). Ο έλεγχος για την ύπαρξη μηνυμάτων πραγματοποιείται με τη βοήθεια της ιδιότητας num_rows, του αντικειμένου $entries. Στη γραμμή 90, εφόσον υπάρχουν μηνύματα και με τη βοήθεια της ίδιας ιδιότητας, ξεκινά η εκτέλεση ενός βρόχου, ο οποίος εμφανίζει τα καταχωρημένα μηνύματα. Για την ανάκτηση των στοιχείων κάθε μηνύματος χρησιμοποιείται η μέθοδος fetch_object(). Αυτή επιστρέφει ένα αντικείμενο, με ιδιότητες τα ονόματα των στηλών (τα πεδία) του πίνακα των αποτελεσμάτων. Η τιμή καθεμίας από αυτές τις ιδιότητες αποτελεί τα δεδομένα της ομώνυμης στήλης (του ομώνυμου πεδίου).

Στις γραμμές 92 έως 94 τυπώνονται τα δεδομένα, ενώ γίνεται και χρήση της συνάρτησης stripslashes(). Η εν λόγω συνάρτηση αφαιρεί τα slashes που προστέθηκαν με την addslashes(), κατά την αρχική αποθήκευση των δεδομένων στη βάση. Ολοκληρώνοντας τον κώδικα PHP της εφαρμογής μας, αποδεσμεύουμε (καταστρέφουμε) το αντικείμενο με τα αποτελέσματα (γραμμή 97) και πραγματοποιούμε μια αποσύνδεση από τη βάση (γραμμή 98).



Η νέα έκδοση της εφαρμογής μας δεν διαφέρει καθόλου ως προς την εμφάνιση σε σχέση με την προηγούμενη. Ο κώδικας όμως έχει αλλάξει σε μεγάλο βαθμό και πλέον αξιοποιεί τις δυνατότητες της MySQL. Έτσι, στο μέλλον θα μπορούμε να αναπτύξουμε την εφαρμογή μας με μοναδικό περιορισμό τη φαντασία μας ;)

αντιγραφή

Skill: Beginner
Tags: rsync, SSH, deltas, backup, encryption

by GiannouG



Ουσιαστική γνωριμία με μια άριστη λύση backup!

Αν είχαμε 1 ευρώ για κάθε αρχείο που χάναμε από λάθος, οι περισσότεροι αυτή τη στιγμή θα είμασταν μεσοπέλαγα πάνω στο καταμαράν μας και θα πίναμε μοχίτο (ή πορτοκαλάδα) ξέγνοιαστοι. Απ' την άλλη, μπορεί να είμαστε αρκετά τυχεροί και να μην έχουμε χάσει κανένα αρχείο από λάθος. Ίσως όμως να έχουμε χάσει δεδομένα ή δουλειά επειδή κάναμε κάποιες τροποποιήσεις (π.χ., σε αρχεία κειμένου ή κώδικα) σβήνοντας τα παλιά αρχεία, τα οποία μετά από λίγο καιρό καταλάβαμε ότι ακόμη χρειαζόμαστε...

Λύσεις για προβλήματα σαν αυτά υπάρχουν πολλές, αλλά δύσκολα κάποια καλύπτει *όλες* τις περιπτώσεις. Σε σχετικό άρθρο του deltaHacker 009 είχαμε ασφαλίσει τα πολύτιμα αρχεία μας καταφεύγοντας στο duplicity και κάνοντας rolling backups με κρυπτογράφηση. Το εν λόγω πρόγραμμα, αν και αρκετά απλό στην χρήση του, προσθέτει πολλές επιπλέον άχρηστες λειτουργίες ή τουλάχιστον λειτουργίες όχι τόσο ενδιαφέρουσες για το σκοπό μας. Πολύ συχνά, βλέπετε, θέλουμε αυτό: Έχουμε έναν φάκελο κι επιθυμούμε να υπάρχει *πάντα* κάπου αντεγραμμένος, χωρίς να μας ενδιαφέρουν τόσο οι παλιότερες εκδόσεις ή το privacy. Θέλουμε, δηλαδή, κάτι σαν "δεύτερο αντίγραφο".



Ο καθένας κρύβει τον θησαυρό του διαφορετικά, αλλά πάνω-κάτω όλοι τις ίδιες τακτικές ακολουθούμε. Σ' αυτό το άρθρο θα δούμε το rsync και πώς αυτό μπορεί να διαφυλάξει εύκολα, γρήγορα και αποτελεσματικά τα δεδομένα μας ;)

rsync to the rescue!

Μπορεί να μη σας έπεισαν οι δικαιολογίες μας, αλλά για εμάς ήταν αρκετές ώστε να μας κάνουν ν' ασχοληθούμε με το rsync — κάτι που θέλαμε για πολύ καιρό τώρα, εδώ που τα λέμε. Σίγουρα οι περισσότεροι έχουμε συναντήσει ξανά το rsync κατά καιρούς. Πιθανότατα, όμως, όπως συνέβαινε και με τον γράφοντα, δεν πείσαμε εαυτούς να το ψάξουμε ούτε θεωρητικά, ούτε και πρακτικά. Αλλά αρκετά με τις εισαγωγές. Τι είναι το rsync;

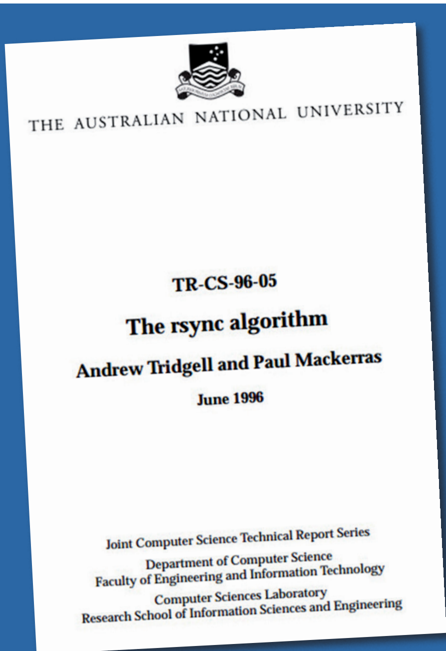


Κοντεύει 18 χρονών και αν ήταν άνθρωπος τώρα θα πήγαινε στην τελευταία τάξη του Λυκείου. Το rsync έχει αλλάξει αρκετά από την πρώτη του έκδοση, μολαταύτα παραμένει το ίδιο χρήσιμο — και δεν έχει σημασία το πόσο έχουν αλλάξει οι ανάγκες και τα συστήματά μας ;)

Πρόκειται για έναν αλγόριθμο, ο οποίος είναι υλοποιημένος σε πρόγραμμα αλλά και σε βιβλιοθήκη, που χρησιμοποιεί έξυπνους τρόπους[1] για να αποφύγει τις άσκοπες μεταφορές δεδομένων. Σε περίπτωση που θελήσουμε να αντιγράψουμε κάποια αρχεία από έναν φάκελο σε άλλο με το rsync, υπάρχουν δύο πιθανότητες: α) το αρχείο δεν θα βρεθεί στον φάκελο-προορισμού και θα αντιγραφτεί, β) στον φάκελο προορισμού θα βρεθεί παλιότερη/τροποποιημένη έκδοση κι έτσι θα "περάσουν" μονάχα οι αλλαγές. Να συμβεί κάτι άλλο, αποκλείεται. Φυσικά, με τα ανάλογα flags κατά την εκτέλεση του rsync, μπορούμε να τροποποιήσουμε την λειτουργία του. Περισσότερα επ' αυτού όμως σε λίγο.

Πρωτεργάτης του rsync είναι ο αυστραλός Andrew Tridgell, γνωστός κι ως ο άνθρωπος που ανέπτυξε τη SAMBA[2]. Αν και ο Tridgel ξεκίνησε να δουλεύει πάνω στο rsync το 1996, στο πλαίσιο του διδακτορικού του, ακόμη και σήμερα το rsync αποτελεί βασικό εργαλείο στις μεταφορές αρχείων μέσω δικτύου κι όχι μόνο. Ο αλγόριθμος

1. Layman's terms για να εξηγήσουμε το delta (Δ) encoding :D
2. Η εκδοχή Free Software του ιδιόκτητου πρωτοκόλλου SMB/CIFS, για τη διαμοίραση αρχείων.



THE AUSTRALIAN NATIONAL UNIVERSITY

TR-CS-96-05

The rsync algorithm

Andrew Tridgell and Paul Mackerras

June 1996

Joint Computer Science Technical Report Series
Department of Computer Science
Faculty of Engineering and Information Technology
Computer Sciences Laboratory
Research School of Information Sciences and Engineering

Όλα ξεκίνησαν από μια εργασία διδακτορικού δύο φοιτητών, με σκοπό να λύσουν ένα πολύ απλό πρόβλημα: Αυτό της πιο αποδοτικής μεταφοράς αρχείων!

του rsync συναντάται σε αρκετά προγράμματα που χρησιμοποιούμε καθημερινά. Είναι κάτι σαν αφανής ήρωας, που υποστηρίζει προϊόντα και υπηρεσίες μεταφοράς αρχείων όπως, π.χ., είναι αυτή του Dropbox! Παρόμοια λογική χρησιμοποιείται και σε άλλα πρωτόκολλα και υπηρεσίες, όπως στο Git.

Τοπικά backups, εύκολα και γρήγορα

Ας ξεκινήσουμε με τα βασικά. Χρησιμοποιούμε Ubuntu Linux3 και θέλουμε τον φάκελο Projects, ο οποίος βρίσκεται στον home directory, να τον αντιγράψουμε σε έναν εξωτερικό σκληρό δίσκο. Για αρχή θα δουλέψουμε στο τερματικό και, μόλις σιγουρευτούμε πως όλα πάνε καλά, θα μεταφέρουμε την εντολή στο cron ώστε να τρέχει αυτόματα, χωρίς εμάς. Πρώτα απ' όλα, όμως, θα χρειαστεί να εγκαταστήσουμε το ίδιο το rsync:

```
$ sudo apt-get update
$ sudo apt-get install rsync
```

Δίνουμε σε ένα νέο τερματικό τις δύο παραπάνω κλασικές, πλέον, εντολές και μόλις εκτελεστεί και η δεύτερη είμαστε έτοιμοι να ξεκινήσουμε. Για αρχή, όπως αναφέραμε και πριν, θα παίξουμε τοπικά. Στόχος μας είναι ο φάκελος /home/deltahacker/Projects (το deltahacker είναι το username του χρήστη μας) να υπάρχει ως αντίγραφο σε κάποιον εξωτερικό σκληρό/thumb drive. Με τη χρήση που κάνουμε, υπολογίσαμε ότι αντίγραφα κάθε μια μέρα είναι αρκετά για να έχουμε πάντα πρόσφατες εκδόσεις των project μας. Ας δούμε, λοιπόν, τη μαγεία4 του rsync:

```
$ rsync -av --delete /home/deltahacker/Projects /media/deltahacker/
MyFlashDisk
```

Αν περιμένατε κάτι περισσότερο, θα σας απογοητεύσουμε. Ας δούμε πιο αναλυτικά τι τρέξαμε. Δύο είναι οι βασικές παράμετροι που χρησιμοποιούμε εδώ: το -a και το --delete. Η παράμετρος –v σημαίνει verbose output, δηλαδή να βλέπουμε αναλυτικά τι λειτουργίες εκτελεί το πρόγραμμα. Φυσικά δεν είναι απαραίτητη και τη χρησιμοποιούμε για να δούμε πώς λειτουργεί το rsync και μόνο.

Η παράμετρος -a βάζει το rsync να δουλέψει σε "archive mode". Η εν λόγω λειτουργία δεν είναι τίποτα παραπάνω από συνδυασμό κάποιων άλλων παραμέτρων του rsync. Συγκεκριμένα, η παράμετρος -a είναι το ίδιο σαν να χρησιμοποιούμε τις παραμέτρους -rlptgoD. Ας δούμε λίγο πιο αναλυτικά τι κάνει κάθε παράμετρος για να καταλάβουμε γιατί τις χρειαζόμαστε.

---

3. Για το άρθρο, μόνο: Τα ίδια βήματα κι εντολές θα δουλέψουν σε όποιο λειτουργικό υπάρχει το rsync.
4. Αν καταφέρετε να κρατήσετε αντίγραφο φακέλων που θα κοιτάει *μόνο* τις αλλαγές και προσθήκες στα αρχεία, και όλα αυτά σε 1 (μια) μόλις γραμμή τερματικού θα χαρούμε να το ακούσουμε :D

-r	από τον αρχικό φάκελο θα μπει σε όλους τους υποφακέλους και θα αντιγράψει όλα τα αρχεία που θα βρει μέσα σε αυτούς
-l	αντιγραφή και των symbolic links που μπορεί να έχουμε δημιουργήσει
-p	διατήρηση δικαιωμάτων χρήστη για τα αρχεία και τους φακέλους
-t	διατήρηση του modification time, δηλαδή του χρόνου επεξεργασίας ενός αρχείου
-g	ομοίως με το -p, αλλά για τα δικαιώματα των χρηστών του group
-o	διατήρηση ownership των αρχείων
-D	αντιγραφή συσκευών συστήματος και άλλων ειδικών αρχείων (π.χ., socket)
--delete	σβήνει τα αρχεία απ' τον φάκελο του backup που *δεν* υπάρχουν στον αρχικό μας φάκελο (π.χ., αρχεία που σβήσαμε εμείς πριν τρέξει το rsync)

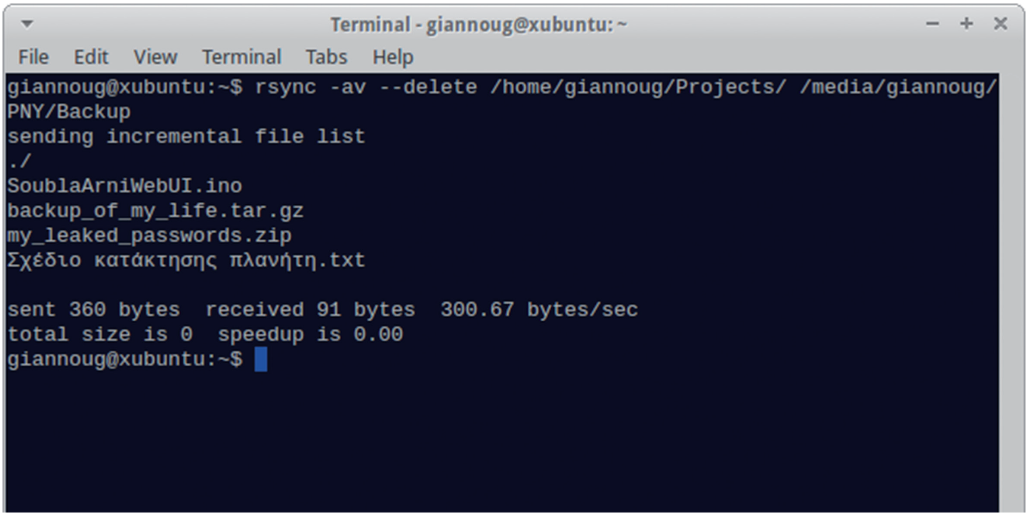
Αν εξαιρέσουμε την τελευταία παράμετρο, η οποία για την δική μας περίπτωση είναι λίγο άχρηστη, τα υπόλοιπα θα μπορούσαμε να πούμε πως είναι απαραίτητα για την σωστή λειτουργία των αντιγράφων μας. Εδώ, να σημειώσουμε πως οι παράμετροι -o και -D δεν λειτουργούν αν το rsync δεν τρέξει με δικαιώματα root. Αυτό δε μας πειράζει καθόλου, μια και τα αρχεία που σκοπεύουμε να αντιγράψουμε είναι δικά μας αρχεία κι έχουμε και δικαιοδοσία υπερχρήστη. Επίσης, δεν πρόκειται να αντιγράψουμε συσκευές Linux, ούτε κάποιο socket. Κοιτώντας και παίζοντας με τις παραμέτρους καταλήξαμε σε ένα συμπέρασμα: το rsync είναι ένα cp on steroids[5] :D

Απομακρυσμένα backups, αλλά το ίδιο εύκολα

Ευτυχώς για εμάς, το rsync υποστηρίζει το πρωτόκολλο SSH για τη μεταφορά των αρχείων σε άλλα μηχανήματα. Χρησιμοποιώντας το SSH, εκτός από την ευκολία που μας δίνει μια και κάθε μηχάνημα που σέβεται τον εαυτό του τρέχει έναν SSH server, μας προσφέρει και την ισχυρή κρυπτογράφηση του ίδιου του πρωτοκόλλου.

Σε αυτό το σημείο να αναφέρουμε πως το rsync υποστηρίζει και λειτουργία σε daemon mode. Στην ουσία, αυτό που κάνει είναι να σηκώνει έναν server και να περιμένει συνδέσεις από κάποιον απομακρυσμένο υπολογιστή. Με μια πρώτη σκέψη είναι αυτό που χρειαζόμαστε — σωστά; Θα σηκώσουμε έναν daemon στον server που θέλουμε να στέλνουμε τα backups μας και, τέλος, αυτό είναι. Υπάρχει όμως ένα μικρό πρόβλημα. Ο daemon αυτός δεν είναι και *τόσο* ασφαλής. Για αρχή, δεν

5. Κι ομολογούμε ότι αυτό πολύ μας αρέσει!



Με μια απλή εντολή μιας γραμμής, μπορούμε να αντιγράψουμε ολόκληρο φάκελο σε πιο ασφαλές μέρος ;) Μην σας παραξενεύουν τα αρχεία, είναι απλά για τις δοκιμές μας...

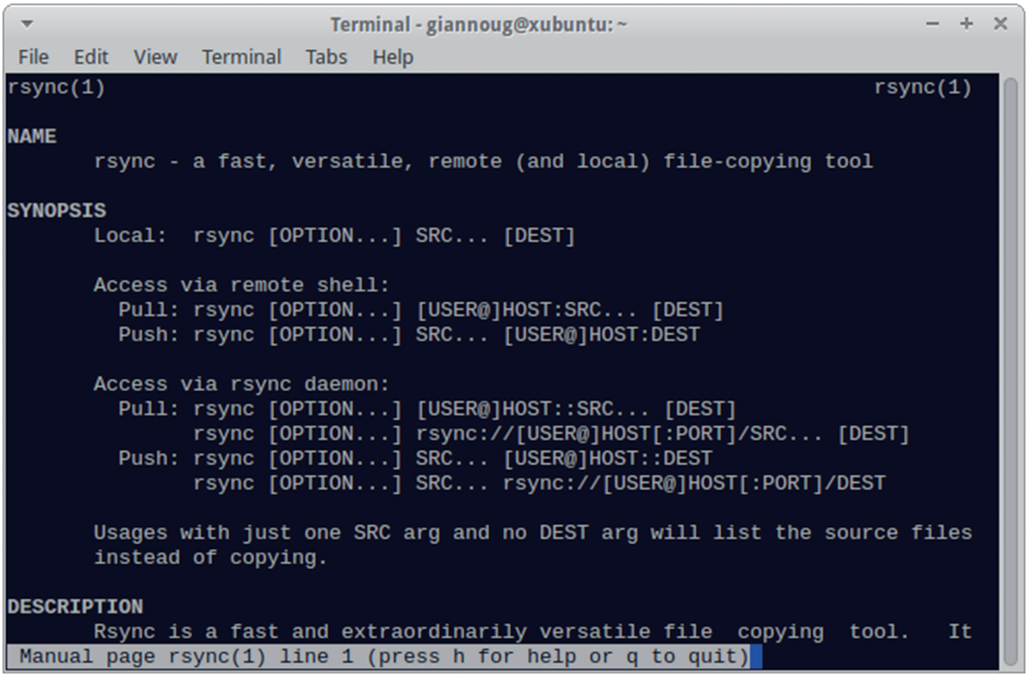
υποστηρίζει κρυπτογράφηση στα δεδομένα που μεταφέρονται, οπότε η χρήση του προτείνεται μονάχα σε τοπικά μηχανήματα. Ο μόνος λόγος που θα ήθελε κάποιος να χρησιμοποιήσει το rsync σε daemon mode είναι αν δεν υπάρχει κάποιος SSH server εγκατεστημένος. Φανταζόμαστε ότι σε embedded συστήματα (π.χ., routers) κάτι τέτοιο είναι αρκετά συχνό φαινόμενο, παρόλο που μετά το deltaHacker 015 τα δικά μας routerάκια τρέχουν, μεταξύ άλλων υπηρεσιών, και SSH :D

Η μόνη αλλαγή που θα χρειαστεί να κάνουμε στις εντολές που είδαμε πριν, είναι να προσθέσουμε μια ακόμη παράμετρο για την ενεργοποίηση του SSH. Επιπλέον, θα χρειαστεί να ορίσουμε το username του χρήστη στον λογαριασμό του οποίου θα συνδεθούμε, αλλά και το hostname του απομακρυσμένου μηχανήματος.

```
$ rsync -azv --delete -e ssh /home/deltahacker/Projects \
dh@pwner.deltahacker.gr:/home/deltahacker/Backup
```

Με την παράμετρο -e δηλώνουμε πως θέλουμε να γίνει μεταφορά των δεδομένων μέσω SSH. Η δεύτερη αλλαγή που κάναμε είναι να ορίσουμε το πού θέλουμε να αντιγράφονται τα αρχεία. Δηλώσαμε το χρήστη αλλά και το hostname του μηχανήματος, ακολουθούμενο από το path που επιθυμούμε να αποθηκεύσουμε τα αρχεία, όπως και πριν. Επιπλέον, προσθέσαμε ακόμη μια παράμετρο για τη συμπίεση των δεδομένων μας κατά την μεταφορά. Το SSH πιθανότατα να κάνει από μόνο του κάποιου είδους συμπίεση, αλλά δεν χάνουμε τίποτα να συμπιέσουμε κι εμείς, πρώτοι, τα δεδομένα μας. Να τονίσουμε ότι η συμπίεση γίνεται μόνο για την μεταφορά των δεδομένων κι όχι για τα τελικά αρχεία!

Τέλος, καλό θα ήταν να δημιουργήσουμε τα απαραίτητα κλειδιά SSH (ακριβέστερα, ένα ζεύγος ιδιωτικού-δημοσίου κλειδιού) ώστε να μη χρειάζεται να γράφουμε συνέχεια τον κωδικό μας προκειμένου να συνδεόμαστε μέσω SSH. Αυτό θα μας χρειαστεί και στην συνέχεια, όταν αυτοματοποιήσουμε τελείως τα backup μας. Η διαδικασία



Τα man pages του rsync κρύβουν πολλές πληροφορίες για την λειτουργία του και βοήθησαν αρκετά στο να ανακαλύψουμε το θαυμάσιο αυτό εργαλείο ;) Καλό θα ήταν να του ρίξετε μια ματιά, πληκτρολογώντας man rsync στο τερματικό.

είναι αρκετά απλή. Ξεκινάμε αμέσως δίνοντας στο σύστημά μας (από εκεί που θέλουμε να αντιγράφουμε τα αρχεία) αυτό:

```
$ ssh-keygen -t rsa
```

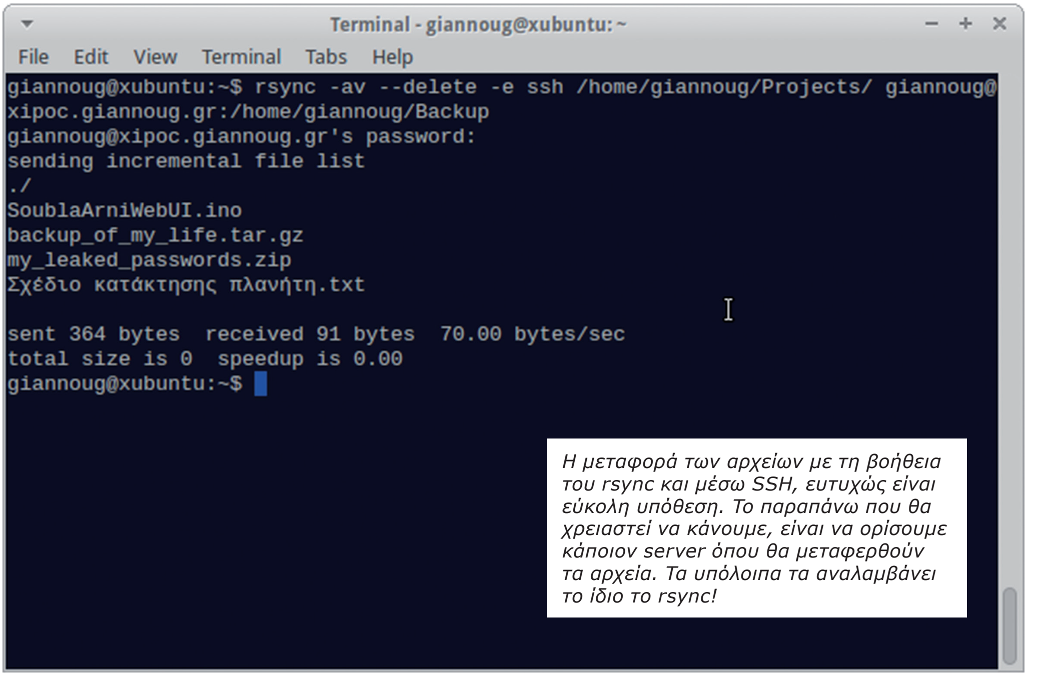
Το πρόγραμμα θα μας ρωτήσει πού θέλουμε να αποθηκεύτει το νέο μας ιδιωτικό κλειδί, αλλά κι αν θέλουμε να το προστατεύσουμε με κάποιο passphrase. Μπορούμε ελεύθερα να πατήσουμε [Enter] για να αφήσουμε τις προεπιλεγμένες ρυθμίσεις. Φυσικά και δεν θέλουμε passphrase για το ιδιωτικό κλειδί, μια κι ο σκοπός που δημιουργούμε το ζεύγος κλειδιών είναι για να συνδεόμαστε χωρίς κανέναν κωδικό (θέλουμε, δηλαδή, να μη χρειάζεται παρέμβαση από πλευράς του χρήστη). Τέλος, θα χρειαστεί να αντιγράψουμε το νέο μας *δημόσιο* κλειδί στον server όπου θα συνδεόμαστε.

```
$ ssh-copy-id dh@pwner.deltahacker.gr
```

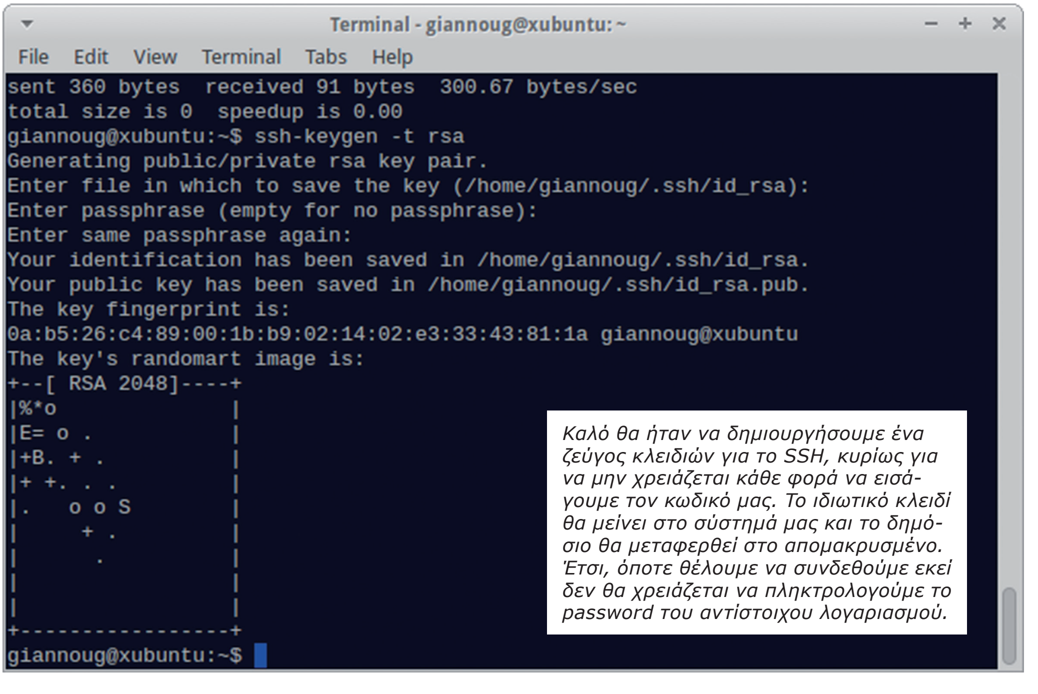
Αφού δώσουμε για τελευταία φορά τον κωδικό μας, το δημόσιο κλειδί θα μεταφερθεί στον server. Είμαστε πανέτοιμοι! Αν ξανατρέξουμε την εντολή, δεν θα μας ζητηθεί κωδικός και η αντιγραφή θα γίνει χωρίς την παρέμβασή μας.

Υπηρεσίες backup, με την ευγενική χορηγία του rsync

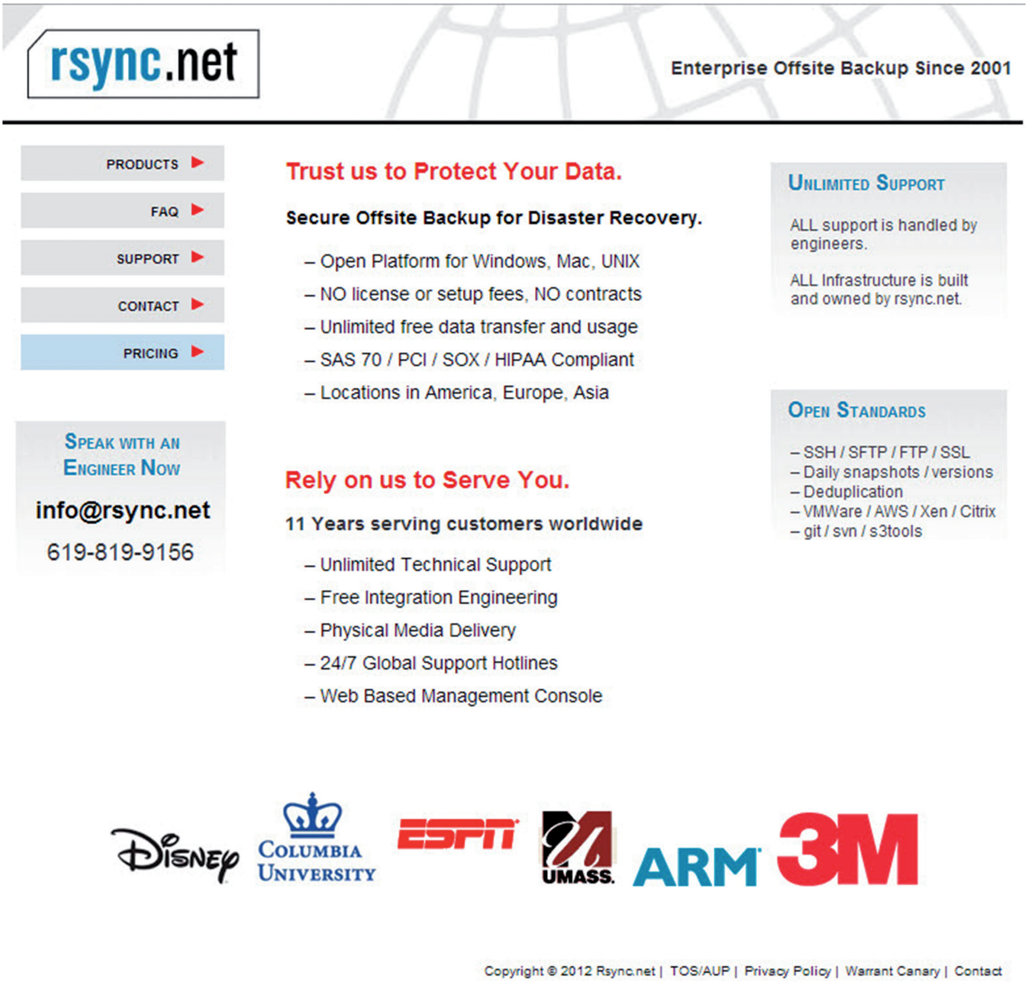
Όλα ωραία, όλα καλά. Τα δεδομένα μας είναι ασφαλή σε κάποιον άλλο σκληρό μας ή/και σε κάποιο άλλο μηχάνημα που εμείς ελέγχουμε. Τι γίνεται όμως σε περίπτωση που χρειαστούμε κάποιο χώρο για την αποθήκευση των δεδομένων μας, αλλά δεν έχουμε το κατάλληλο hardware ή απλά δεν θέλουμε να μπλέξουμε με το στήσιμο



Η μεταφορά των αρχείων με τη βοήθεια του rsync και μέσω SSH, ευτυχώς είναι εύκολη υπόθεση. Το παραπάνω που θα χρειαστεί να κάνουμε, είναι να ορίσουμε κάποιον server όπου θα μεταφερθούν τα αρχεία. Τα υπόλοιπα τα αναλαμβάνει το ίδιο το rsync!



Καλό θα ήταν να δημιουργήσουμε ένα ζεύγος κλειδιών για το SSH, κυρίως για να μην χρειάζεται κάθε φορά να εισάγουμε τον κωδικό μας. Το ιδιωτικό κλειδί θα μείνει στο σύστημά μας και το δημόσιο θα μεταφερθεί στο απομακρυσμένο. Έτσι, όποτε θέλουμε να συνδεθούμε εκεί δεν θα χρειάζεται να πληκτρολογούμε το password του αντίστοιχου λογαριασμού.



Σε περίπτωση που θελήσουμε κάποιο χώρο για να φυλάξουμε τα δεδομένα μας με περισσότερη ασφάλεια, μπορούμε να στραφούμε σε κάποιον provider όπως το rsync.net.

των servers; Αν σας απασχολεί αυτό το ερώτημα, τότε σίγουρα οφείλετε να σκεφτείτε κι αυτό: Πρακτικά εξ ορισμού, τα δεδομένα μας να είναι τόσο σημαντικά για εμάς που σίγουρα δε θέλουμε να διακινδυνεύσουμε να τα χάσουμε, οπότε θα θελήσουμε να ψάξουμε για κάτι σίγουρο και σταθερό.

Ευτυχώς για εμάς, υπάρχουν υπηρεσίες που προσφέρουν χώρο αποθήκευσης για τα πολύτιμα δεδομένα μας χρησιμοποιώντας το πρωτόκολλο του rsync. Οι τιμές διαφέρουν αρκετά από εταιρεία σε εταιρεία, αλλά αυτό μάλλον έχει να κάνει με την ασφάλεια που προσφέρει ο κάθε πάροχος. Για παράδειγμα, στη μικρή μας έρευνα εντοπίσαμε πάροχο ο οποίος, αν του ζητηθεί, ταχυδρομεί ακόμη και τον φυσικό δίσκο όπου αποθηκεύονταν τα backup μας! Αν πάλι αναρωτιέστε ποιος είναι ο πιο δημοφιλής και πιο παλιός πάροχος για rsync backups —και πιθανότητα η αιτία που τα ξεκίνησε όλα— η απάντηση είναι το rsync.net ;)

Εν κατακλείδι...

Ούτε κρυπτογράφηση, ούτε περίεργα κλειδιά, ούτε μπέρδεματα, ούτε προβλήματα. Το rsync είναι ίσως το παλιότερο εργαλείο αντιγραφής[6] αρχείων και, όπως αποδεικνύεται από τη δοκιμασία του χρόνου, ένα από τα καλύτερα. Μπορούμε επίσης να χρησιμοποιούμε archives για τα backup μας. Αναλυτικότερα, θα μπορούσαμε να φτιάχνουμε πρώτα ένα αρχείο ZIP ή tar.bz2 με τα περιεχόμενα του φακέλου μας και να στέλνουμε *αυτό* το αρχείο μέσω rsync. Με κάθε κλήση του rsync θα μεταφέρεται *μόνο* το κομμάτι του συμπιεσμένου archive που έχει αλλάξει. Με την ίδια λογική μπορούμε να χρησιμοποιούμε *και* Truecrypt containers. Μάλιστα, οι ίδιοι οι δημιουργοί του Truecrypt έχουν επιβεβαιώσει ότι η φύση του Truecrypt δεν εμποδίζει[7] το rsync να δουλεύει αποδοτικά.

Υπάρχουν περιπτώσεις που απλά θέλεις ένας φάκελος να *μη χαθεί* και το rsync θα φροντίσει για ακριβώς αυτό! Φυσικά, γράφοντας πιο πολύπλοκα shell scripts μπορούμε να κάνουμε πολλά περισσότερα πράγματα, όπως, π.χ., να κρατάμε incremental backups ή/και να κρυπτογραφούμε. Αλλά αυτά για την ώρα δεν μας απασχολούν. Εξάλλου, το duplicity που ρυθμίσαμε στο deltaHacker 009 λειτουργεί ακόμη ;)

Encrypted Filesystem Images with TrueCrypt

TrueCrypt is a widely used, open source disk encryption software that allows you to create a large, single, encrypted file that can be mounted as a drive letter and used like any other filesystem.

When the filesystem is unmounted, the encrypted filesystem reverts back to its single, encrypted file.

We are happy to report that the internal structure of truecrypt disk images make it possible to efficiently rsync them to a remote location. This means that after the initial (complete) upload of your TrueCrypt disk image, subsequent uploads will efficiently transfer only the changes that have been made to the encrypted filesystem since its last upload.

*Το rsync υποστηρίζει αποδοτικά *και* Truecrypt containers (σχεδόν πάντα). Θα συμφωνήσετε, φανταζόμαστε, ότι εδώ μιλάμε για ένα ονειρικό πάντρεμα αποδοτικής λύσης backup και πανίσχυρης λύσης κρυπτογραφίας!*

6. Για την ακρίβεια, όπως είδαμε, κάνει κάτι παραπάνω και πιο ενδιαφέρον από μια *απλή* αντιγραφή

7. Αν το container αλλάζει τελείως μορφή κάθε φορά που αλλάζουμε δεδομένα, τότε το rsync δε θα βρίσκει διαφορές και θα μεταφέρει συνέχεια το ίδιο αρχείο από την αρχή!


delta HACKER
Στη 3η χρονιά και συνεχίζουμε ακάθεκτοι!
Μηνιαίο περιοδικό deltaHacker, http://deltahacker.gr
Ethical Hacking / Δίκτυα / Ασφάλεια / Προγραμματισμός / Ηλεκτρονικά
twitter.com/deltaHacker
facebook.com/deltaHacker

Skill: Intermediate
Tags: session, authentication, LAMP, PHP, MySQL, web, development



Web development σε περιβάλλον LAMP [Μέρος 8]

Όσον αφορά στην ανάπτυξη web εφαρμογών σε περιβάλλον LAMP stack, οι ορίζοντές μας έχουν διευρυνθεί αρκετά. Είμαστε πλέον στην ευχάριστη θέση να αναπτύσσουμε εφαρμογές οι οποίες βασίζονται στη λογική του συνδυασμού των επιμέρους τμημάτων που αποτελούν το LAMP stack. Στο παρόν άρθρο, τελευταίο της σειράς μας, θα εξετάσουμε ένα κοινότοπο χαρακτηριστικό των web εφαρμογών, το οποίο δεν είναι άλλο από την ταυτοποίηση των χρηστών. Θα δούμε, φυσικά, πώς υλοποιούμε τους σχετικούς μηχανισμούς στα δικά μας προγράμματα.

by multiPetros

Το πρωτόκολλο HTTP έχει σαν αποστολή τη μεταφορά κειμένου μεταξύ των υπολογιστών ενός δικτύου. Η επιτυχία του σε αυτή την ομολογουμένως ταπεινή αλλά *εξαιρετικά* σημαντική αποστολή, είναι τεράστια κι αδιαμφισβήτητη. Για πολλούς, η επιτυχία οφείλεται στην απλότητα του πρωτοκόλλου. Όπως όμως θα γνωρίζετε, η απλότητα δεν είναι πάντοτε ευεργετική. Υπάρχουν φορές που λειτουργεί και σαν εμπόδιο ή, τέλος πάντων, ως μια πρόσθετη δυσκολία…

Είναι, βλέπετε, που το HTTP αδιαφορεί πλήρως για την κατάσταση στην οποία βρίσκεται μια σύνδεση. Για την ακρίβεια, ούτε καν προβλέπει την έννοια της κατάστασης και γι' αυτό χαρακτηρίζεται κι ως stateless protocol. Με απλά λόγια, δεν παρέχει κανέναν εγγενή μηχανισμό για τον έλεγχο του αν ένα σύνολο αιτημάτων αποτελούν αυτοτελή οντότητα, αν ανήκουν σε μια "ομάδα", ή όχι. Για να κατανοήσετε τι σημαίνει αυτό, σκεφτείτε κάποια πρωτόκολλα που ανήκουν στην αντίθετη κατηγορία (stateful), όπως τα SSH, FTP και Telnet. Με αυτά τα πρωτόκολλα, μετά από ένα επιτυχημένο login, μπορούμε να στέλνουμε όσα αιτήματα θέλουμε κι ο διακομιστής είναι βέβαιο ότι θα τα εκτελεί *χωρίς* να ζητάει ξανά και ξανά τα στοιχεία μας. Με τη βοήθεια αυτών των πρωτοκόλλων, ο εκάστοτε διακομιστής μπορεί να γνωρίζει αν τα αιτήματα που λαμβάνει προέρχονται από τον ίδιο χρήστη κι αν τελικά ανήκουν στην ίδια "ομάδα" ή όχι. Το HTTP δεν προβλέπει τίποτα σχετικό!

Παρ' όλα αυτά, η συντριπτική πλειονότητα των εφαρμογών του ιστού βασίζονται σε κάποιο είδος επαλήθευσης της ταυτότητας των χρηστών. Έτσι, μπορούμε να κάνουμε login σε κάποια εφαρμογή web και στη συνέχεια να αποκτήσουμε πρόσβαση σε κλειδωμένο περιεχόμενο και υπηρεσίες, να αποθηκεύσουμε τις προτιμήσεις μας κ.ο.κ. Αναρωτηθήκατε ποτέ πώς γίνεται αυτό;

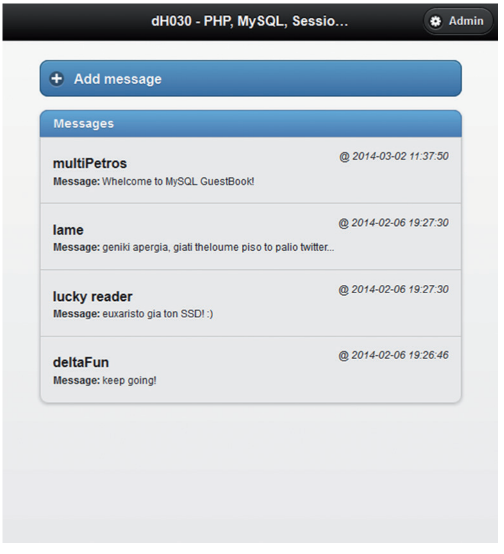
Μια λύση έχει δώσει η ομάδα ανάπτυξης της PHP, μάλιστα χωρίς την παραμικρή παρέκκλιση από της προδιαγραφές του πρωτοκόλλου HTTP. Για την αντιμετώπιση του προβλήματος, εισήγαγαν την έννοια των συνεδριών (sessions). Κάθε συνεδρία διαθέτει ένα προστατευμένο σύνολο μεταβλητών, καθώς κι ένα κρυπτογραφημένο κλειδί. Οι μεταβλητές περιέχουν τα δεδομένα που προορίζονται για τον εκάστοτε "νόμιμο" χρήστη ή, τέλος πάντων, καθορίζουν το αν και πού θα έχει πρόσβαση ο εκάστοτε χρήστης. Το κρυπτογραφημένο κλειδί, από την άλλη, λειτουργεί ως διαβατήριο εισόδου. Ο χρήστης που διαθέτει το κλειδί μιας συνεδρίας έχει και πρόσβαση στις αντίστοιχες προστατευμένες μεταβλητές.

Κουλουράκια, αλλά όχι του καφέ

Όπως αναφέραμε, απαραίτητο συστατικό των συνεδριών της PHP αποτελούν τα κρυπτογραφημένα κλειδιά. Αυτά τα μικρά πακέτα δεδομένων αποστέλλονται στον browser του χρήστη μέσα σε cookies ή μέσω των URLs. Από προεπιλογή χρησιμοποιούνται τα cookies — και μόνον αν διαπιστωθεί ότι είναι απενεργοποιημένα γίνεται χρήση των URLs. Τα cookies, βλέπετε, παρέχουν μεγαλύτερη ασφάλεια, αφού ένα URL θα μπορούσε ευκολότερα να εκτεθεί σε αδιάκριτα βλέματα. Τι είναι όμως αυτά τα cookies που ακούμε ξανά και ξανά; Τα cookies αποτελούν πνευματικό παιδί της Netscape και η χρήση τους προτάθηκε και υλοποιήθηκε για πρώτη φορά το 1995, στον ομώνυμο browser (Netscape Navigator). Την αρχαία αυτή εποχή, τα cookies είχαν ως αποστολή την αποθήκευση πληροφοριών στον υπολογιστή του χρήστη με στόχο την αναγνώρισή τους από τον κώδικα που τρέχει στον εκάστοτε διακομιστή και την παροχή προσωποποιημένου περιεχομένου. Όπως αντιλαμβάνεστε, μιλάμε για κάτι άκρως πρωτοποριακό, τουλάχιστον για τα δεδομένα της εποχής. Τα cookies,

λοιπόν, είναι πολύ μικρά αρχεία κειμένου, τα οποία διατηρεί ο browser σε κάποια περιοχή του δίσκου μας. Στις μέρες μας δεν υπάρχει κανένας browser που να μην υποστηρίζει τα cookies.

Ένα cookie δημιουργείται με την αποστολή των κατάλληλων πληροφοριών από τον server προς τον browser του χρήστη, μέσα από τα headers ενός αιτήματος HTTP. Μάλιστα, ο ορισμός ενός cookie πρέπει να προηγείται οποιουδήποτε άλλου header. Αυτό αποτελεί μια κρίσιμη λεπτομέρεια, που πρέπει να έχει κατά νου οποιοσδήποτε θέλει να ορίσει "χειροκίνητα" τα δικά του cookies. Ευτυχώς για εμάς, η PHP αναλαμβάνει να λύσει όλα τα ζητήματα χρονισμού. Από μέρους μας, αρκεί να χρησιμοποιήσουμε τον επίσημο τρόπο δημιουργίας συνεδριών, που προβλέπει η PHP.



Η κεντρική οθόνη της εφαρμογής μας, όπως παρουσιάζεται στους απλούς χρήστες. Η μόνη διαφορά στην εμφάνισή της, σε σχέση με την προηγούμενη έκδοση, εντοπίζεται στην προσθήκη του εικονιδίου σύνδεσης, στην πάνω δεξιά γωνία.

Δημιουργία συνεδριών

Η έναρξη μιας συνεδρίας με την PHP αποτελεί εξαιρετικά απλή υπόθεση. Αρκεί να χρησιμοποιήσουμε τη συνάρτηση session_start(). Βέβαια, μιας κι αναφερθήκαμε ήδη στη σημασία που έχει ο χρόνος δημιουργίας κι αποστολής των cookies, πρέπει να υπογραμμίσουμε μια σημαντική λεπτομέρεια: Η προαναφερθείσα συνάρτηση πρέπει να τοποθετείται *οπωσδήποτε* στην αρχή του προγράμματος! Ας δούμε όμως τι κάνει για εμάς η session_start. Αρχικά, ελέγχει αν υπάρχει ήδη κάποια ενεργή συνεδρία. Εάν εντοπιστεί κάποια, φορτώνονται οι αντίστοιχες μεταβλητές στον πίνακα $_SESSION. Από εκεί μπορούμε να διαβάσουμε το περιεχόμενό τους, αλλά και να τις τροποποιήσουμε. Στην περίπτωση που δεν υπάρχει ενεργή συνεδρία, δημιουργείται μια νέα, με έναν κενό πίνακα μεταβλητών.

Οι μεταβλητές που δηλώνουμε στον πίνακα $_SESSION παραμένουν ενεργές και είναι προσβάσιμες από οποιοδήποτε τμήμα κώδικα μέσα στην εφαρμογή μας, μέχρι τη στιγμή που θα τερματίσουμε τη σύνοδο. Φυσικά, μπορούμε να διαγράψουμε οποιαδήποτε μεταβλητή θέλουμε από μια συνεδρία, όποια στιγμή το αποφασίσουμε. Κάτι τέτοιο γίνεται με τη βοήθεια της unset και κάπως έτσι: unset($_SESSION['variable_name']). Παρεμπιπτόντως, για τον έλεγχο ύπαρξης μιας μεταβλητής προσφέρεται η συνάρτηση isset, η οποία ακολουθεί παρόμοιο συντακτικό. Ο τερματισμός μιας συνεδρίας επιτυγχάνεται με τη συνάρτηση session_destroy(). Η εν λόγω συνάρτηση απενεργοποιεί την ενεργή σύνοδο και διαγράφει αυτόματα όλες τις μεταβλητές του πίνακα $_SESSION.

Πριν προχωρήσουμε, αξίζει να σταθούμε σε μια σπαζοκεφαλιά που αντιμετωπίζουν αρκετοί προγραμματιστές, κατά τα πρώτα τους βήματα με την PHP. Όπως είπαμε, η συνάρτηση session_start() πρέπει να τοποθετείται υποχρεωτικά στην αρχή του

εκάστοτε προγράμματος. Παρ' όλα αυτά, υπάρχουν περιπτώσεις που ενώ έχουμε τοποθετήσει τη συνάρτηση στη σωστή θέση, λαμβάνουμε ένα μήνυμα σφάλματος του στιλ "ERROR Warning: session_start() [function.session-start]: Cannot send session cookie – headers already sent by...". Ο συνήθης ύποπτος για αυτό το σφάλμα είναι η κωδικοποίηση του αρχείου με τον κώδικα. Τα προγράμματα σε PHP είναι φρόνιμο να τα γράφουμε σε Unicode *χωρίς* BOM[1]. Αν συναντήσετε σφάλματα όπως το παραπάνω, ενώ η συνάρτηση session_start βρίσκεται στη σωστή θέση, ανοίξτε τον αγαπημένο σας text editor και ελέγξτε τις ρυθμίσεις του.

Στην πράξη...

Η θεωρητική προσέγγιση του ζητήματος των συνεδριών είναι μάλλον περίπλοκη κι ενδέχεται να σας έχουμε κουράσει, αν όχι και μπερδέψει. Γι' αυτό το λόγο θα μελετήσουμε ένα μικρό παράδειγμα χρήσης των συνεδριών, το οποίο εκτείνεται σε τρία επιμέρους προγράμματα.

```
<?php
session_start();

if(isset($_POST['mag'])) {
    $_SESSION['mag_name'] = $_POST['mag'];
    echo 'Magazine name stored!';
} else {
?>
    <form action="store.php" method="post">
        <label for="mag">Magazine</label>
        <input type="text" name="mag" value="<?php
            if(isset($_SESSION['mag_name']))
                echo $_SESSION['mag_name'];
            else
                $_SESSION['mag_name'] = "";
            ?>">
        <input type="submit">
    </form>
<?php
}
echo '<p><a href="display.php">Show best magazine</a> | <a href="logout.php">Logout</a></p>';
?>
```

1. Το BOM προέρχεται από τα αρχικά των λέξεων Byte Order Mark.

Το πρόγραμμα ξεκινά με την έναρξη της συνεδρίας, κάνοντας χρήση της συνάρτησης session_start(). Κατόπιν, με τη βοήθεια της isset, ελέγχουμε αν έχει σταλεί κάποια τιμή για τη μεταβλητή mag, με τη μέθοδο POST. Αν έχει σταλεί κάποια τιμή, την αποθηκεύουμε στη μεταβλητή της συνεδρίας mag_name και τυπώνουμε ένα σχετικό μήνυμα (γραμμές 05 και 06). Αν δεν έχουν ληφθεί δεδομένα με τη μέθοδο POST, τυπώνουμε μια μικρή φόρμα (γραμμές 09 έως 18). Η φόρμα ζητάει από το χρήστη να δώσει το όνομα του αγαπημένου του περιοδικού. Στη γραμμή 12 —και κάπου στη μέση του κώδικα της φόρμας— ελέγχουμε αν υπάρχει ήδη η μεταβλητή συνεδρίας με το όνομα mag_name. Αν ο χρήστης έχει υποβάλλει ξανά τη φόρμα, η μεταβλητή θα έχει οριστεί ήδη. Σε αυτή την περίπτωση, θέλουμε η τιμή της μεταβλητής να εμφανίζεται μέσα στο text field, ως προεπιλογή. Μετά τη φόρμα, τυπώνουμε δύο συνδέσμους. Ο πρώτος παραπέμπει στη σελίδα display.php, όπου εμφανίζεται το καλύτερο περιοδικό του κόσμου, ενώ η δεύτερη παραπέμπει στη σελίδα logout.php, όπου τερματίζεται η συνεδρία και, ουσιαστικά, πραγματοποιείται μια αποσύνδεση. Ας συνεχίσουμε τώρα τη μελέτη του παραδείγματος, εξετάζοντας τον κώδικα της σελίδας display.php.

```
01 <?php
02 session_start();
03 if(!isset($_SESSION['mag_name'])) {
04 echo 'I don\'t know... It\'s because you are not logged in.<br><a
   href="store.php">Login now</a>';
05 exit();
06 }
07 echo $_SESSION['mag_name'].' is the best magazine!';
08 echo '<p><a href="store.php">Set another</a> | <a href="logout.
   php">Logout</a></p>';
09 ?>
```

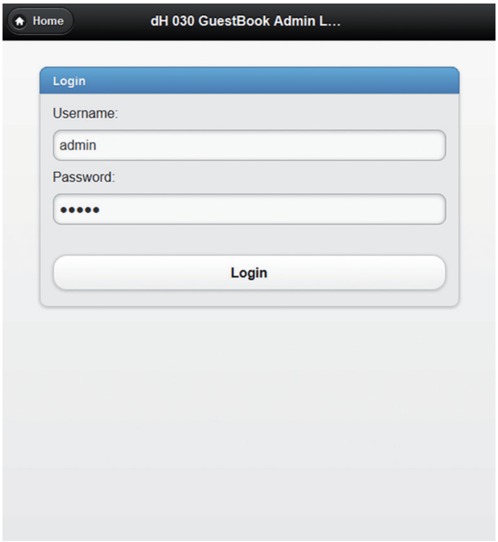
Αποστολή του κώδικα που περιλαμβάνεται σε αυτό το αρχείο, είναι η εμφάνιση της τιμής της μεταβλητής mag_name. Όπως βλέπετε, ο κώδικας ξεκινά και πάλι με μια κλήση στη συνάρτηση session_start(). Αμέσως μετά ελέγχουμε αν έχει οριστεί η μεταβλητή mag_name, με τη βοήθεια της isset(). Αν δεν υπάρχει η μεταβλητή, ο κώδικας συμπεραίνει ότι δεν έχουμε συνδεθεί και τυπώνει το σχετικό μήνυμα. Στη γραμμή 05 εκτελείται και η συνάρτηση exit(), η οποία διακόπτει την εκτέλεση του κώδικα PHP. Αν η μεταβλητή mag_name έχει οριστεί, οι γραμμές 04 και 05 δεν θα εκτελεστούν και το πρόγραμμα θα προχωρήσει στη γραμμή 07. Όπως είναι πολύ εύκολο να καταλάβετε, από εκεί και κάτω τυπώνεται η τιμή της μεταβλητής μας, μαζί με το απαραίτητο συνοδευτικό κειμενάκι. Ας δούμε τώρα και τον κώδικα της σελίδας logout.php.

```
<?php
session_start();
session_destroy();
?>
<h1>You are now logged out!</h1>
<p><a href="display.php">Show my favorite magazine…</a> | <a href="favorite_mag.php">Login</a></p>
```

Εδώ το τμήμα του κώδικα PHP είναι πάρα πολύ μικρό. Ξεκινάμε με την κλήση της session_start(), ώστε να δηλώσουμε στην PHP ότι *και* αυτό το πρόγραμμα ανήκει στη σύνοδο. Αμέσως μετά καλούμε τη συνάρτηση session_destroy(), με την οποία ζητάμε από την PHP να τερματίσει τη σύνοδο και να διαγράψει τις σχετικές μεταβλητές. Αυτό ήταν όλο! Ακολούθως, εμφανίζουμε το κατάλληλο μήνυμα στο χρήστη και τυπώνουμε δυο συνδέσμους προς τις άλλες σελίδες της εφαρμογής μας. Όπως μπορείτε εύκολα να καταλάβετε, η σελίδα favorite_mag.php περιλαμβάνει τον κώδικα που εξετάσαμε στο πρώτο μέρος του παραδείγματος.

Αναβαθμισμένο GuestBook

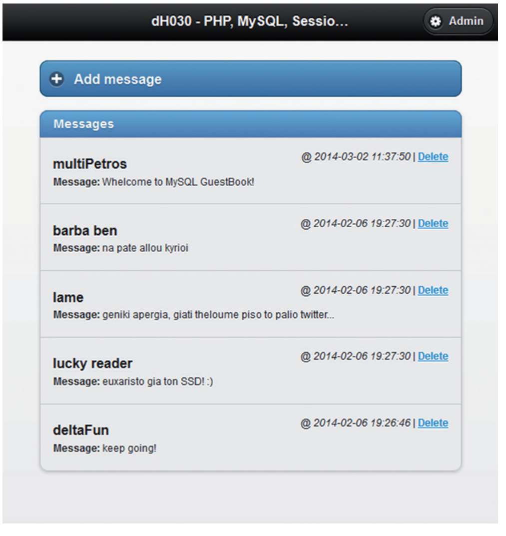
Οι συνεδρίες λειτουργίας είναι εξαιρετικά χρήσιμες —για να μην πούμε απαραίτητες— κατά την ανάπτυξη εφαρμογών web. Μετά και από το μικρό παράδειγμα που εξετάσαμε, θέλουμε να πιστεύουμε ότι έχετε κατανοήσει όλες τις βασικές αρχές λειτουργίας των συνεδριών. Κάπως έτσι, μπορούμε να προχωρήσουμε σε μια πιο σύνθετη εφαρμογή: Στην αξιοποίηση των συνεδριών για την αναβάθμιση της εφαρμογής GuestBook. Στην πιο πρόσφατη έκδοση της εφαρμογής μας, οι χρήστες μπορούν να αναρτούν όποιο μήνυμα θέλουν κι αυτό αποθηκεύεται αυτόματα σε μία βάση δεδομένων. Δυστυχώς, εκτός από τους φιλήσυχους χρήστες που θέλουν απλά να κάνουν τη δουλειά της, υπάρχουν και οι κακόβουλοι, όπως και τα ενοχλητικά spam bots. Με την τρέχουσα υλοποίηση της εφαρμογής μας, είναι αδύνατο να αποφύγουμε τα ανεπιθύμητα μηνύματα. Ο μόνος τρόπος για να απαλλαγούμε από αυτά είναι να συνδεθούμε στο MySQL Monitor και να τα διαγράψουμε χειροκίνητα, χρησιμοποιώντας τα κατάλληλα ερωτήματα. Όπως αντιλαμβάνεστε, κάτι τέτοιο είναι τουλάχιστον απαράδεκτο τη σήμερον ημέρα. Κά-



Η οθόνη σύνδεσης των διαχειριστών. Η εμφάνιση κι ο τρόπος χειρισμού δεν διαφέρει σε τίποτα απ' αυτά που συναντάμε στις κανονικές εφαρμογές web…

πως έτσι μπαίνουν στο παιχνίδι οι συνεδρίες, με τη βοήθεια των οποίων θα αναπτύξουμε ένα σύστημα πιστοποίησης των χρηστών. Αξιοποιώντας αυτό το σύστημα θα μπορούμε να εμφανίζουμε πρόσθετες επιλογές, όπως αυτές της διαγραφής, της απόκρυψης ή της τροποποίησης ενός μηνύματος, μόνο σε κάποιον συγκεκριμένο χρήστη, τον οποίο θα θεωρούμε διαχειριστή.

Ως συνήθως, θα παρουσιάσουμε τον κώδικα της εφαρμογής τμηματικά κι εστιάζοντας μόνο στα κρίσιμα σημεία. Εσείς, φυσικά, μπορείτε να μελετήσετε και να πειραματιστείτε με ολόκληρο τον κώδικα της αναβαθμισμένης εφαρμογής, αφού τον κατεβάσετε από το `http://bit.ly/dh030guestbook`.



Η αρχική οθόνη, όπως εμφανίζεται σε κάποιον διαχειριστή. Παρατηρείστε το σύνδεσμο delete, που εμφανίζεται δίπλα στην ημερομηνία δημιουργίας κάθε μηνύματος.

Αλλαγές στη βάση

Πριν κάνουμε οτιδήποτε άλλο, θα χρειαστεί να τροποποιήσουμε ελαφρώς τη βάση δεδομένων μας. Διώξτε μακριά τον πονοκέφαλο που απειλεί να σας κυριεύσει: Έναν πίνακα θα προσθέσουμε μόνο, με το όνομα admin. Ο νέος πίνακας θα διαθέτει μόλις δύο πεδία: Ένα με το όνομα usr και ένα ακόμα με το όνομα pwd. Όπως μπορείτε να φανταστείτε, σε αυτόν τον πίνακα θα φυλάσσονται τα στοιχεία σύνδεσης (όνομα χρήστη και συνθηματικό) των χρηστών, οι οποίοι θα έχουνε δικαιώματα διαχειριστή επί της εφαρμογής μας. Σαν πρωτεύον κλειδί θα επιλέξουμε εκείνο που θα φιλοξενεί τα ονόματα, εφόσον, ούτως ή άλλως, κάθε χρήστης-διαχειριστής θα έχει διαφορετικό όνομα. Γι' αυτές τις εργασίες ανοίγουμε το MySQL Monitor κι εκτελούμε το ακόλουθο ερώτημα SQL:

```
create table admin (
	usr char(50) not null primary key,
	pwd char(32) not null
);
```

Για το μήκος του πεδίου pwd επιλέξαμε τους 32 χαρακτήρες, καθώς τόσο είναι και το μήκος των hashes που παράγει ο αλγόριθμος MD5. Άλλωστε, δεν σκοπεύουμε να αποθηκεύουμε τους κωδικούς πρόσβασης σε plain text, αλλά σε μια μορφή που η ανάκτησή τους θα είναι αδύνατη ή, τέλος πάντων, εξαιρετικά δύσκολη! Βλέπετε, η "αντιστροφή" του αλγόριθμου MD5 είναι πρακτικά αδύνατη. Επομένως, ακόμη κι αν καταφέρει κανείς να κλέψει τα MD5 hashes, μόνο με τη μέθοδο του brute force (και με απίστευτη τύχη ή άπειρη υπομονή) θα μπορέσει να ανακτήσει τους κωδικούς. Σε αυτό το σημείο ενδέχεται να προκύψει ένα εύλογο ερώτημα, σε όσους ασχολούνται

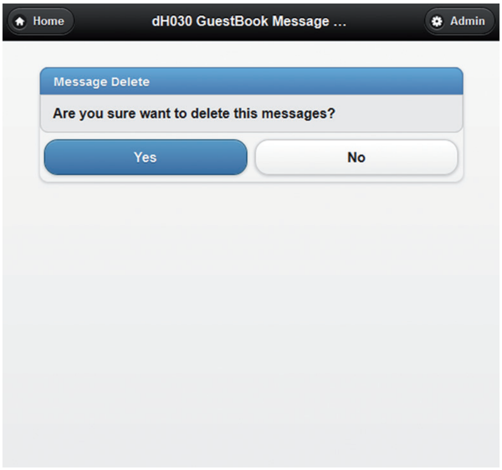
για πρώτη φορά με αυτά τα ζητήματα: Εφόσον είναι πρακτικά αδύνατο να πάρουμε τον κωδικό, όταν γνωρίζουμε μόνο το MD5 hash, πώς θα επαληθεύουμε ότι ο κωδικός που εισάγει κάθε φορά ένας επισκέπτης είναι σωστός; Πολύ απλά, κάθε φορά που ένας επισκέπτης προσπαθεί να συνδεθεί, θα κωδικοποιούμε το συνθηματικό που εισάγει κατά MD5 και θα το συγκρίνουμε με το αποθηκευμένο. Αν τα δύο hashes είναι ίδια, τότε ο επισκέπτης έδωσε τον σωστό κωδικό. Η χρήση των hashes για την αποθήκευση των κωδικών αποτελεί μια καλή και αναγκαία πρακτική. Σας συνιστούμε να την ακολουθείτε σε όλες τις εφαρμογές σας.

Ας ορίσουμε τώρα και έναν διαχειριστή, με το ευφάνταστο όνομα admin και τον υπερασφαλή κωδικό 12345. Για την εισαγωγή των στοιχείων στο σχετικό πίνακα, αρκεί να εκτελέσουμε το ακόλουθο ερώτημα:

```
insert into admin values ('admin',md5('12345'));
```

Αλλαγές στον κώδικα

Σε ό,τι αφορά στον κώδικα της ανανεωμένης εφαρμογής, πραγματοποιούμε ορισμένες γενναίες προσθήκες καθώς και κάποιες μικρές τροποποιήσεις. Κατ' αρχάς, δημιουργούμε το αρχείο structure.php, στο οποίο ορίζουμε τις συναρτήσεις print_header() και print_footer(). Όπως καταλαβαίνετε, αυτές οι δύο συναρτήσεις θα είναι υπεύθυνες για τη σχεδίαση της επικεφαλίδας και του κατώτατου τμήματος των σελίδων μας. Διατηρώντας αυτές τις συναρτήσεις σε ένα ξεχωριστό αρχείο, θα μπορούμε να αλλάζουμε την αισθητική των σελίδων της εφαρμογής μας επεμβαίνοντας σε μία μόνο θέση. Αυτό κρίθηκε απαραίτητο επειδή η εφαρμογή μας θα αποτελείται πλέον από περισσότερες της μιας σελίδες.



Με την επιλογή του συνδέσμου διαγραφής, η εφαρμογή δεν προχωρά άμεσα στη διαγραφή, αλλά ζητά την επιβεβαίωσή μας.

Μπορείτε να βρείτε το structure.php στο πακέτο με τον πηγαίο κώδικα. Ο κώδικάς του δεν θα μας απασχολήσει περισσότερο, καθώς δεν παρουσιάζει ιδιαίτερο ενδιαφέρον, ούτε περιέχει κάτι αξιοπρόσεκτο. Έχοντας ξεμπερδέψει με το εν λόγω αρχείο, ξεκινήσαμε τις επεμβάσεις στο αρχείο της αρχικής σελίδας (index.php). Εκεί αντικαταστήσαμε τα τμήματα κώδικα που σχεδιάζουν την επικεφαλίδα και την κατώτατη περιοχή της σελίδας, με κλήσεις προς τις αντίστοιχες συναρτήσεις του structure.php. Φυσικά, πριν καλέσουμε τις συναρτήσεις, φροντίσαμε για την ενσωμάτωση του σχετικού αρχείου:

```
include('structure.php')
```

Η παραπάνω γραμμή τοποθετήθηκε ακριβώς κάτω από την κλήση της session_start(). Η επόμενη αλλαγή που κάναμε, εντοπίζεται στις γραμμές 76 έως 80:

```
076 if(isset($_SESSION['loggedin']) && $_SESSION['loggedin'] == true)
077    $deltxt = " | <a href=\"delete.php?&id=".strval($entry->id) .
    "\">Delete</a>";
078 else
079    $deltxt = "";
080 echo "<li><p class=\"ui-li-aside ui-li-desc\"><em>@
    ".stripslashes($entry->date)." </em>".$deltxt."</p>";
```

Όπως βλέπετε, στη γραμμή 76 ελέγχουμε αν έχει οριστεί η μεταβλητή συνεδρίας loggedin, καθώς και το περιεχόμενό της. Η μεταβλητή loggedin είναι αυτή που αποφασίσαμε ότι θα σηματοδοτεί τη σύνδεση κάποιου χρήστη με δικαιώματα διαχείρισης. Αν λοιπόν έχει οριστεί αυτή η μεταβλητή κι έχει την τιμή true, δημιουργούμε έναν σύνδεσμο προς το delete.php και τον αποθηκεύουμε στη μεταβλητή $deltxt. Το URL του συνδέσμου ενσωματώνει και το id του τρέχοντος μηνύματος, με τη μέθοδο GET. Όπως καταλαβαίνετε, ο κώδικας της σελίδας delete.php εκτελεί τη λειτουργία της διαγραφής του μηνύματος με το επιλεγμένο id. Ας επιστρέψουμε στον κώδικα, όμως. Στην περίπτωση που δεν έχει συνδεθεί κάποιος διαχειριστής (γραμμή 078), αφήνουμε το περιεχόμενο της $deltxt κενό. Τέλος, στη γραμμή 80 τυπώνουμε το περιεχόμενο της $deltxt αμέσως μετά την ημερομηνία του εκάστοτε μηνύματος. Έτσι, αν ο χρήστης έχει συνδεθεί σαν διαχειριστής, θα βλέπει μαζί με την ημερομηνία κι έναν σύνδεσμο για τη διαγραφή του αντίστοιχου μηνύματος.

Διαγραφή μηνυμάτων

Μετά από τις παραπάνω επεμβάσεις, δημιουργήσαμε το αρχείο delete.php, του οποίου ο κώδικας αναλαμβάνει τη διαγραφή των μηνυμάτων:

```
<?php
if (isset($_SESSION['loggedin']) && $_SESSION['loggedin'] == true) {
if (isset($_GET['sure']))
    $sure = (trim($_GET['sure']) == '1') ? true : false;
else
    $sure = null;
$id = (isset($_GET['id']) == true) ? trim($_GET['id']) : "";
if ($sure === null) {
?>
    <li>Are you sure want to delete this message?</li>
```

```
    <fieldset class="ui-grid-a">
        <div class="ui-block-a">
            <a href="delete.php?id=<?php echo $id ?>&sure=1" data-role="button" data-ajax="false" data-theme="b">Yes</a>
        </div>
        <div class="ui-block-b">
            <a href="delete.php?id=<?php echo $id ?>&sure=0" data-role="button" data-ajax="false">No</a>
        </div>
    </fieldset>
<?php
}
elseif ($sure === true) {
      $db = new mysqli('localhost', 'gbook_web', 'securepass', 'gbook');
      $entries = $db->query("delete from messages where id=$id");
 ?>
      <li>Done! Message deleted.</li>
      <a href="index.php" data-role="button" data-ajax="false">Go Home</a>
<?php
} else {
?>
      <li>Canceled!</li>
      <a href="index.php" data-role="button" data-ajax="false">Go Home</a>
<?php
}
} else {
?>
<li>You are not an administrator. Keep out!</li>
<a href="index.php" data-role="button" data-ajax="false">Go Home</a>
```

Πριν προβούμε σε οποιαδήποτε ενέργεια, εξετάζουμε αν έχει γίνει login από κάποιον διαχειριστή. Στη συνέχεια ελέγχουμε αν έχουμε λάβει τη μεταβλητή sure με τη μέθοδο GET και, αν ναι, τσεκάρουμε την τιμή της. Η μεταβλητή sure χρησιμοποιείται ως ένα πρόσθετο μέτρο ασφάλειας, για να αποφύγουμε την ακούσια διαγραφή μηνυμάτων. Όταν ο διαχειριστής φτάνει στη σελίδα delete.php από το σύνδεσμο δίπλα σε κάποιο μήνυμα, η μεταβλητή sure δεν έχει τιμή. Ο κώδικας το αντιλαμβάνεται και πριν προχωρήσει στη διαγραφή του μηνύματος ζητά από το χρήστη μια επιβεβαίωση. Ανάλογα με την απάντηση του χρήστη φορτώνουμε εκ νέου τη σελίδα, περνώντας με τη μέθοδο GET την κατάλληλη τιμή για τη μεταβλητή sure (1 αν ο χρήστης απάντησε καταφατικά και 0 αν απάντησε αρνητικά). Μαζί με την τιμή για τη μεταβλητή sure, στέλνουμε εκ νέου και το id του προς διαγραφή μηνύματος. Έτσι, τη δεύτερη φορά που φορτώνεται η σελίδα, ο κώδικας έχει στη διάθεσή του το id του μηνύματος καθώς και την επιβεβαίωση ή την άρνηση του διαχειριστή. Αν η τιμή της μεταβλητής sure είναι 1, το πρόγραμμα προχωρά στη διαγραφή του μηνύματος, εκτελώντας το κατάλληλο ερώτημα τροποποίησης του πίνακα messages. Όπως αντιλαμβάνεστε, αν θέλαμε να δημιουργήσουμε μια πραγματικά άψογη εφαρμογή, ο κώδικας δεν θα προχωρούσε αμέσως στη διαγραφή. Θα φρόντιζε προηγουμένως να τσεκάρει αν το δοσμένο id είναι έγκυρο. Επιπρόσθετα, θα ήλεγχε και την έκβαση της εκτέλεσης του ερωτήματος. Όλα αυτά όμως αποτελούν μια απλή άσκηση, που αποφασίσαμε να αφήσουμε σ' εσάς :)

Διαδικασία σύνδεσης

Μέχρι στιγμής έχουμε εξετάσει τις αλλαγές και τις προσθήκες που απαιτούνται, ώστε η εφαρμογή μας να διαγράφει μηνύματα κατ' εντολή ενός διαχειριστή. Αυτό που δεν έχουμε δει, είναι το πώς θα πραγματοποιείται η σύνδεση (login) και η έναρξη μιας αντίστοιχης συνεδρίας. Όλα αυτά πραγματοποιούνται στο αρχείο admin.php:

```
if(isset($_GET['action']) && trim(htmlspecialchars($_GET['action']) == 'logout')) {
session_destroy();
?>
<li>Your session is terminated. You have logged out</li>
<a href="admin.php" data-role="button" data-ajax="false">Login again</a>
<?php
} elseif (isset($_GET['action']) && trim(htmlspecialchars($_GET['action']) == 'login')) {
  $usr = (isset($_POST['usr']) == true) ? addslashes(trim(htmlspecialchars($_POST['usr']))) : "";
  $pwd = (isset($_POST['pwd']) == true) ? addslashes(trim(htmlspecialchars($_POST['pwd']))) : "";
  $db = new mysqli('localhost', 'gbook_web', 'securepass', 'gbook');
```

```
  $entries = $db->query("select * from admin where usr='$usr' and pwd=md5('$pwd')");

  if($entries->num_rows < 1) {
     $_SESSION['loggedin'] = false;
  }
  else{
     $_SESSION['loggedin'] = true;
     $_SESSION['usr'] = $usr;
  }
  if ($_SESSION['loggedin'] == true) {
?>
     <li>You have logged in successfully!</li>
     <a href="index.php" data-role="button" data-ajax="false">Go Home</a>
  <?php
  } else {
  ?>
     <li>Wrong name or password!</li>
     <a href="admin.php" data-role="button" data-ajax="false">Try again...</a>
  <?php
  }
} else {
if(isset($_SESSION['loggedin']) && $_SESSION['loggedin'] == 'true') {
?>
   <li>You have allready logged in, as <?php echo $_SESSION['usr']; ?></li>

  <fieldset class="ui-grid-a">
     <div class="ui-block-a">
```

```
            <a href="index.php" data-role="button" data-ajax="false">Go Home</a>
        </div>
        <div class="ui-block-b">
            <a href="admin.php?action=logout" data-role="button" data-ajax="false" data-theme="b">Log out</a>
        </div>
    </fieldset>
    <?php
}
else {
?>
        <li>
            <form name="login" action="admin.php?action=login" method="post" data-ajax="false">
                <label for="usr">Username:</label>
                <input name="usr" id="usr" type="text" maxlength="50" placeholder="Your Username">
                <label for="pwd">Password:</label>
                <input name="pwd" id="pwd" type="password" maxlength="50" placeholder="Your Password">
                <label> </label><input type="submit" value="Login">
            </form>
        </li>
<?php
}
}
```

Ο κώδικας ξεκινάει ελέγχοντας αν έχει σταλεί η μεταβλητή action με τη μέθοδο GET. Η τιμή της εν λόγω μεταβλητής δηλώνει την ενέργεια που θέλουμε να πραγματοποιηθεί. Αν η τιμή της είναι "logout", ο κώδικας τερματίζει την τρέχουσα συνεδρία και καταστρέφει τις μεταβλητές της καλώντας τη συνάρτηση session_destroy(). Στην περίπτωση που η μεταβλητή action έχει την τιμή "login", ο κώδικας ελέγχει αν έχουν σταλεί οι μεταβλητές usr και pwd, μέσω της μεθόδου POST. Αυτό βέβαια (η αποστολή της action με τιμή login καθώς και η αποστολή των usr και pwd) πραγμα-

τοποιείται μόνο με την υποβολή της φόρμας σύνδεσης. Εφόσον έχουν σταλεί οι δύο μεταβλητές, το πρόγραμμα αντιπαραβάλλει τις τιμές τους με τα στοιχεία στον πίνακα admin. Βέβαια, ειδικά για την τιμή της pwd, στην αντιπαραβολή δεν χρησιμοποιείται η τιμή της μεταβλητής αλλά το αντίστοιχο md5 hash. Αν τα στοιχεία σύνδεσης είναι σωστά, ορίζεται η μεταβλητή συνεδρίας loggedin σε true, όπως επίσης και η μεταβλητή συνεδρίας usr, με το όνομα χρήστη του διαχειριστή. Σε αντίθετη περίπτωση, το πρόγραμμα εμφανίζει στο χρήστη ένα κατάλληλο μήνυμα λάθους.

Ένα σημείο που αξίζει την προσοχή σας αποτελεί η γραμμή 63. Η υποβολή της φόρμας στέλνει τα δεδομένα με χρήση της μεθόδου POST αλλά, ταυτόχρονα, χρησιμοποιώντας ένα κατάλληλα κατασκευασμένο URL, στέλνει και τη μεταβλητή action με τη μέθοδο GET.

Κάπου εδώ ενδέχεται να σκέφτεστε ότι οι έλεγχοι ασφάλειας που πραγματοποιεί η εφαρμογή μας είναι πολύ χαλαροί. Δεν έχετε άδικο. Ο σκοπός μας όμως ήταν η ολοκληρωμένη παρουσίαση μιας εφαρμογής που στηρίζεται σε συνεδρίες κι όχι η κατασκευή μιας ετοιμοπόλεμης εφαρμογής, που θα μπορούσε να αντέξει στις δοκιμασίες του πραγματικού κόσμου.

Σαλπάρουμε για νέες περιπέτειες!

Κάπου εδώ, το ταξίδι μαζί μας στο απέραντο πέλαγος του LAMP Stack, όπως άλλωστε και κάθε ταξίδι, φτάνει προς το τέλος του. Προορισμός και συνάμα στόχος μας δεν ήταν η στενή και στείρα γνώση περί ανάπτυξης web εφαρμογών, αλλά η εκμάθηση όλων των απαραίτητων εργαλείων για την πλεύση σε οποιοδήποτε σημείο του χάρτη τραβά η όρεξή μας. Απαραίτητη προϋπόθεση, φυσικά, είναι η ύπαρξη/ανάπτυξη της περιέργειας και του ενδιαφέροντος για εξερεύνηση και περιπέτεια. Πλέον, η παρουσία μας στην κουβέρτα του πλεούμενού σας δεν κρίνεται απαραίτητη. Πιστός σας σύμμαχος είναι η τεκμηρίωση της PHP, από τον επίσημο δικτυακό τόπο (`www.php.net/manual`), καθώς κι αυτή της MySQL (`https://dev.mysql.com/doc`). Βεβαίως, δεν χρειάζεται να κάνουμε καν λόγο για τη συνεχή εξάσκηση και τον πειραματισμό. Έχετε το πλοίο, κατέχετε και την ναυτοσύνη, αφήστε λοιπόν τη φαντασία και το μεράκι να σας ταξιδέψουν όσο μακριά επιθυμείτε!

Skill: Beginner
Tags: error-based, boolean-based, time-based, union-based, SQL injection, sqlmap, DVWA, stacked queries

Δεν σας προτείνουμε ν' αποδεχτείτε μοιρολατρικά την ιδέα του κινδύνου, ούτε ν' αντιμετωπίσετε το ενδεχόμενο μιας επίθεσης με επιπολαιότητα. Θέλουμε όμως να γνωρίσετε τις μεθόδους και τα όπλα των επιτιθέμενων, ώστε να είσαστε άψογα προετοιμασμένοι *πριν* ακόμα δεχτείτε χτύπημα. Γιατί, πώς να το κάνουμε, η γνώση και η πρόληψη αποτελούν τους σπουδαιότερους συμμάχους σε κάθε μάχη.



by Sp1r@lEv0lut10n

SQL injections: Εξοικειωθείτε με τον κίνδυνο!

Σκεφτείτε για λίγο τον αθλητή μιας μαχητικής τέχνης. Κάποιον άνθρωπο σε προχωρημένο επίπεδο, που συμμετέχει και σε μεγάλες διοργανώσεις. Νομίζετε ότι αδιαφορεί για την έκβαση των αγώνων του; Πιστεύετε ότι αντιμετωπίζει τις συγκρούσεις σαν παιχνιδάκι; Τίποτα από τα δύο: Παίρνει τα πάντα πολύ σοβαρά. Μπορεί να γνωρίζει εκ των προτέρων ότι πρόκειται για ένα άθλημα και είναι βέβαιος ότι οι αντίπαλοι είναι συναθλητές κι όχι εγκληματίες, αλλά δεν αφήνει τίποτα στην τύχη. Πολύ πριν μπει στο ρινγκ, έχει δεχτεί αμέτρητα χτυπήματα από φίλους και προπονητές. Πριν φτάσει να δώσει μια μάχη, έχει μελετήσει την τακτική δεκάδων αντιπάλων κι έχει αναπτύξει μια άμυνα για κάθε επίθεση.

Νομίζουμε ότι καταλαβαίνετε πολύ καλά που το πάμε. Στο προηγούμενο τεύχος αναφερθήκαμε στον κίνδυνο που διατρέχουν οι δικτυακές εφαρμογές από τις επιθέσεις SQL injection. Ένας οποιοσδήποτε χρήστης ενδέχεται να είναι καλοπροαίρετος, αλλά από την άλλη δεν αποκλείεται να είναι και κακόβουλος. Κι επειδή είναι αδύνατον για κάποια δικτυακή εφαρμογή να γνωρίζει τις προθέσεις του επισκέπτη, πρέπει να είναι σχεδιασμένη με μεγάλη προσοχή και να φιλτράρει εξονυχιστικά τις επιθυμίες του εκάστοτε χρήστη. Αλλά αυτά τα συζητήσαμε στο τεύχος 029. Στο παρόν άρθρο θα ξεφύγουμε από τη θεωρητική παρουσίαση του ζητήματος. Θα εξετάσουμε τα διάφορα είδη των επιθέσεων SQL injection, θα γνωρίσουμε ένα απλό αλλά πανίσχυρο όπλο που έχουν στη διάθεσή τους οι επιτιθέμενοι, ενώ θα στήσουμε κι ένα μικρό χώρο προπόνησης: Ένα περιβάλλον μελέτης και εξάσκησης.

Πάντα επίκαιρες

Οι επιθέσεις SQL injection υπάρχουν πολλά χρόνια τώρα και δεν αποτελούν κάτι νέο στο χώρο της Διαδικτυακής ασφάλειας. Μάλιστα, η κεντρική ιδέα πίσω από τον τρόπο λειτουργίας τους είναι γνωστή και μπορεί να γίνει εύκολα κατανοητή. Ωστόσο, σύμφωνα με πολλές μελέτες, οι επιθέσεις SQL injection βρίσκονται στην κορυφή της λίστας με τις πιο διαδεδομένες επιθέσεις. Όσο κι αν σας φαίνεται παράξενο, υπάρχουν προγραμματιστές που αγνοούν τον κίνδυνο, όπως κι άλλοι που τον αντιμετωπίζουν με επιπολαιότητα. Εξάλλου, ακόμη κι όσοι παίρνουν το θέμα στα σοβαρά, δεν παύουν να είναι άνθρωποι κι ως εκ τούτου… επιρρεπείς στο σφάλμα. Όλα αυτά δεν αποτελούν λόγο για να παραιτηθούμε, αλλά ένα σημαντικό κίνητρο για να τις μελετήσουμε όσο περισσότερο γίνεται. Ας ξεκινήσουμε, λοιπόν, εξετάζοντας τα βασικά είδη των SQL injection attacks.

- **Error-based.** Οι εφαρμογές μας δεν πρέπει να φιλτράρουν *μόνο* τα δεδομένα που δέχονται από το χρήστη. Πρέπει να ελέγχουν και τα μηνύματα που δέχονται από τη βάση, όπως κι εκείνα που εμφανίζουν μόνες τους, σύμφωνα με τα δεδομένα που λαμβάνουν από τη βάση. Όταν μια εφαρμογή εμφανίζει ανεξέλεγκτα οτιδήποτε λαμβάνει από τη βάση, ακόμα και τα μηνύματα σφάλματος, ο επιτιθέμενος μπορεί ουσιαστικά να συνομιλήσει με τη βάση. Σχηματίζοντας τα κατάλληλα ερωτήματα, είναι σε θέση να εξαναγκάσει τη βάση να του φανερώσει όλα της τα μυστικά: Από τη δομή των πινάκων, μέχρι τα ονόματα των πεδίων, τον τύπο και το περιεχόμενό τους.
- **Boolean-based blind.** Σε αυτές τις επιθέσεις, η κύρια επιδίωξη του κακόβουλου χρήστη είναι να αντλήσει ευαίσθητες πληροφορίες. Να μάθει, για παράδειγμα, τα ονόματα πινάκων ή των στηλών ενός πίνακα και, γενικότερα, να μελετήσει τη διάρθρωση της βάσης δεδομένων ή ακόμα και τη θέση αρ-

χείων κι άλλων πόρων στον server. Ένα σχετικό SQL query περιλαμβάνει ένα τμήμα, το οποίο είναι γνωστό ότι θα γίνει δεκτό και θα εκτελεστεί με επιτυχία από τον κώδικα της εφαρμογής. Αυτό το "νομότυπο" τμήμα του ερωτήματος, όμως, συνδέεται μέσω μίας λογικής πράξης (AND ή OR) με κάποιο "αδιάκριτο" ερώτημα. Από την απάντηση της εφαρμογής, ο επιτιθέμενος μπορεί να συμπεράνει αν το ερώτημα εκτελέστηκε επιτυχώς ή όχι και κατ' επέκταση μπορεί να διαπιστώσει αν το δεύτερο σκέλος του ερωτήματος είναι αληθές ή όχι. Για παράδειγμα, αν το δεύτερο μέρος του ερωτήματος αφορά στην ύπαρξη μιας στήλης με το τάδε όνομα, ο επιτιθέμενος μπορεί να διαπιστώσει αν υπάρχει στήλη με το συγκεκριμένο όνομα. Με αυτόν τον τρόπο, ο επιτιθέμενος μπορεί να σιγά-σιγά και με πολλές δοκιμές, ν' ανακαλύψει όλα τα ονόματα των στηλών ενός πίνακα.

- **Time-based blind.** Υπάρχουν εφαρμογές που είναι σχεδιασμένες με τέτοιο τρόπο, ώστε τα δεδομένα που εμφανίζουν στο χρήστη (τα μηνύματα που επιστρέφουν), να μη μαρτυρούν την έκβαση της εκτέλεσης των ερωτημάτων — τουλάχιστον όχι πάντα. Σε αυτές τις περιπτώσεις, οι επιθέσεις που αναφέραμε προηγουμένως είναι από δύσκολες έως αδύνατες. Ωστόσο, οι επιτιθέμενοι διατηρούν τη δυνατότητα να αντλήσουν πληροφορίες με πλάγιο τρόπο. Τα σχετικά ερωτήματα περιλαμβάνουν και πάλι έναν έλεγχο για την ύπαρξη συγκεκριμένων στοιχείων. Ταυτόχρονα, ενσωματώνουν και μια οδηγία προς τη βάση δεδομένων, που επιβάλει την εισαγωγή μιας μικρής καθυστέρησης όταν ο σχετικός έλεγχος είναι επιτυχημένος. Συγκρίνοντας το χρόνο απόκρισης κατά την εκτέλεση ενός νόμιμου ερωτήματος με το χρόνο απόκρισης για την εκτέλεση ενός "εχθρικού" ερωτήματος, ο επιτιθέμενος μπορεί να συμπεράνει την επιτυχία ή την αποτυχία του ελέγχου που περιλαμβάνει το ερώτημά του.

- **UNION query-based.** Εδώ ο επιτιθέμενος προσπαθεί να αντλήσει πληροφορίες από τη βάση δεδομένων με άμεσο τρόπο κι όχι με την εις άτοπο απαγωγή. Τα σχετικά ερωτήματα ενσωματώνουν τον τελεστή UNION ALL, που επιβάλει το συνδυασμό (τη συγχώνευση) των αποτελεσμάτων από δύο ξεχωριστά ερωτήματα SELECT. Αν η εφαρμογή τυπώνει τα δεδομένα που λαμβάνει από τη βάση, είναι πιθανό να τυπώσει κι εκείνα που προέρχονται από ένα παράνομο —ή τέλος πάντων ανεπιθύμητο— ερώτημα SELECT.

- **Stacked queries.** Αυτή η επίθεση είναι άκρως επικίνδυνη! Φανταστείτε δύο ή περισσότερα ερωτήματα, τα οποία χωρίζονται με ελληνικό ερωτηματικό (ο χαρακτήρας τερματισμού των ερωτημάτων) και τα οποία αποστέλλονται "πακέτο" προς τη βάση. Η βάση θα εκτελέσει διαδοχικά όλα τα ερωτήματα, χωρίς να σταματήσει στο πρώτο. Έτσι, αν η εφαρμογή υποστηρίζει την αποστολή πολλαπλών ερωτημάτων[1] ή, πιο απλά, αν δεν ελέγχει τη μορφή του σχηματιζόμενου ερωτήματος, υπάρχει ο κίνδυνος να εκτελέσει μια νόμιμη ενέργεια και ταυτόχρονα μια παράνομη, όπως, π.χ., τη διαγραφή ενός πίνακα, την εκτέλεση μιας συνάρτησης της βάσης δεδομένων που αλληλεπιδρά με το λειτουργικό σύστημα του server και πάει λέγοντας.

1. Πρόκειται για πολύ κακή σχεδιαστική επιλογή. Ευτυχώς, μπορεί πάντα ν' αποφευχθεί με την κατάλληλα σχεδίαση του κώδικα.

Η καλή γνώση των εν δυνάμει απειλών που αντιμετωπίζει μια εφαρμογή αποτελεί απαραίτητο εφόδιο για την αντιμετώπισή τους (με την ίδια λογική που ο εντοπισμός ενός προβλήματος αποτελεί αναγκαία προϋπόθεση για την επίλυσή του). Η επίγνωση του κινδύνου όμως δεν αρκεί. Για να τον αντιμετωπίσουμε, πρέπει να πάρουμε και τα κατάλληλα μέτρα.

Crash test!

Κάθε φορά που ολοκληρώνουμε μία εφαρμογή νιώθουμε ικανοποίηση και μια κάποια υπερηφάνεια. Στο κάτω-κάτω ξοδέψαμε ενέργεια, αξιοποιήσαμε γνώσεις και επιστρατεύσαμε ευφυΐα που δεν έχουν όλοι οι προγραμματιστές — ή τουλάχιστον έτσι νομίζουμε :S Με λίγα λόγια, είναι συχνό φαινόμενο να υποτιμάμε τις γνώσεις και τα εφόδια των επιτιθέμενων. Η πραγματικότητα όμως δεν συμφωνεί πάντοτε μαζί μας. Αν και είναι ελάχιστοι αυτοί που πιστεύουν ότι το δημιούργημά τους είναι ευπαθές, οι επιτυχημένες επιθέσεις ακόμα και σε πανάκριβα, υπερσύγχρονα συστήματα είναι συχνές. Από όλα αυτά προκύπτει αβίαστα το εξής συμπέρασμα: Η υπέρμετρη εμπιστοσύνη στις ικανότητές μας αποτελεί κακό σύμβουλο. Κάθε φορά που ολοκληρώνουμε μια εφαρμογή, λοιπόν, είναι φρόνιμο να την υποβάλλουμε σε πολλούς κι εξονυχιστικούς ελέγχους. Αν θέλετε σκεφτείτε, π.χ., και τους κατασκευαστές αυτοκινήτων, οι οποίοι καταστρέφουν δεκάδες οχήματα προκειμένου να διαπιστώνουν αν πράγματι προσφέρουν την απαιτούμενη ασφάλεια. Μάλλον ξέρουν τι κάνουν, έτσι δεν είναι;

Όπως κατά τη σχεδίαση και την υλοποίηση μιας εφαρμογής, έτσι και για το στάδιο του ελέγχου χρειαζόμαστε καλή γνώση των μεθόδων που αξιοποιούν οι επιτιθέμενοι. Αυτό όμως είναι από δύσκολο κι απαιτεί εμπειρία που δεν μπορεί να διαθέτει ένας αρχάριος. Δεν αποκλείεται εξάλλου να παραβλέψουμε ή και να ξεχάσουμε κάποιον έλεγχο, παρ' όλο που σε γενικές γραμμές τον είχαμε υπόψη. Για παρόμοιους λόγους, ακόμα και οι επαγγελματίες του χώρου, οι λεγόμενοι penetration testers, επιστρατεύουν ειδικά εργαλεία που ενσωματώνουν κι αυτοματοποιούν ένα μεγάλο εύρος επιθέσεων. Ένα τέτοιο εργαλείο, στην περίπτωση των επιθέσεων SQL injection, είναι το sqlmap.

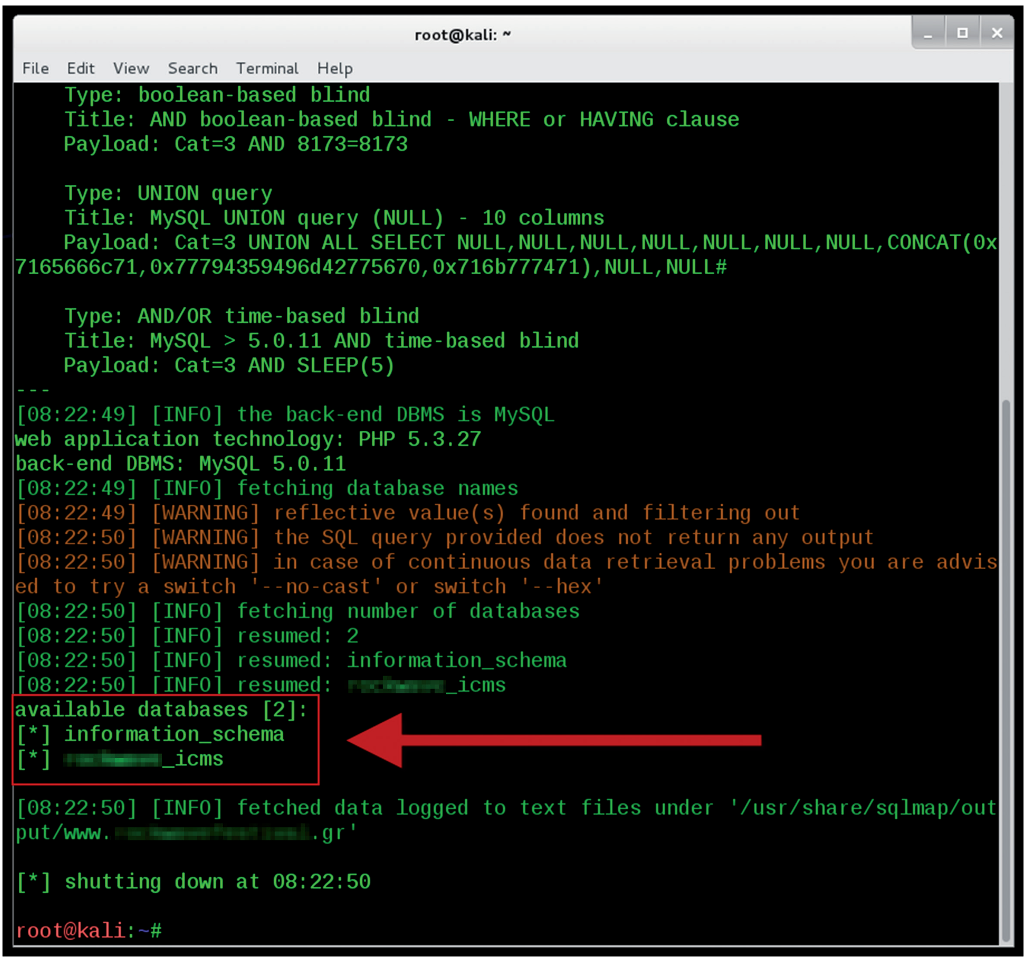
Και κάπου εδώ, φίλες και φίλοι, θα μπορούσε άνετα να ξεκινήσει ένα flame-war. Βλέπετε, υπάρχουν πολλά εργαλεία γι' αυτή τη δουλειά και το καθένα έχει τους υποστηρικτές του. Το sqlmap, όμως, είναι ιδιαίτερα εύχρηστο *και* επεκτάσιμο. Αυτό σημαίνει ότι μπορεί να το αξιοποιήσει εύκολα ένας νέος προγραμματιστής όπως κι ένας πεπειραμένος, που θα έχει τις γνώσεις για να το "εμπλουτίσει" με περισσότερες τεχνικές επιθέσεων. Εξάλλου, η ευχρηστία του sqlmap το καθιστά αγαπημένο εργαλείο πολλών μαθητευόμενων hacker, όπως επίσης και πολλών script kiddies. Δεν θα ήταν καλό αν η εφαρμογή μας αντιμετώπιζε με επιτυχία τις επιθέσεις με χρήση ενός τέτοιου εργαλείου;

Πανίσχυρες επιθέσεις, αλλά με άνεση

Το sqlmap (`http://sqlmap.org`) ενσωματώνει δεκάδες τεχνικές για την ανίχνευση του εκάστοτε συστήματος διαχείρισης βάσεων δεδομένων (MySQL, PostgreSQL, Microsoft, Oracle κ.ά.). Αφού αντλήσει τη συγκεκριμένη πληροφορία από το στόχο, μπορεί να επιστρατεύει άλλες τόσες μεθόδους για να αντλήσει δεδομένα: Ονόματα βάσεων, πινάκων και πεδίων, το είδος των αποθηκευμένων δεδομένων και, φυσικά, τα ίδια τα δεδομένα. Μάλιστα, επειδή είναι γραμμένο στην αγαπημένη μας Python,

μπορεί να τρέξει σε όλα τα λειτουργικά συστήματα. Εμείς το δοκιμάσαμε μέσα από το Kali, τη δημοφιλή διανομή Linux που χρησιμοποιείται για penetration testing.

Το sqlmap δέχεται πάρα πολλές παραμέτρους που ενεργοποιούν τις διάφορες λειτουργίες του. Όμως, για να κάνει οτιδήποτε, πρέπει προηγουμένως να του υποδείξουμε έναν στόχο: Τη διεύθυνση της εφαρμογής στην οποία θα επιτεθεί. Εδώ δεν αρκεί να δώσουμε οποιοδήποτε URL παραπέμπει στο στόχο. Πρέπει να δώσουμε αυτό που μπορεί να χαρακτηριστεί ως attack vector. Το sqlmap μπορεί να επιτεθεί σε εφαρμογές που λαμβάνουν δεδομένα με τη μέθοδο GET, με τη μέθοδο POST ή μέσω cookies. Έτσι, αυτό που ζητά από εμάς είναι ένα URL προς τη σελίδα που εικάζουμε ότι είναι ευπαθής, μαζί με τις μεταβλητές που μεταφέρουν τα δεδομένα του χρήστη. Οι τιμές που θα δώσουμε σε αυτές τις μεταβλητές είναι αδιάφορες. Βλέπετε, το sqlmap θα τις αλλοιώσει ούτως ή άλλως, στην προσπάθεια να πετύχει το SQL injection. Επομένως, από μέρους μας δεν χρειάζεται τίποτα εξειδικευμένο και



Σε μια από τις πρώτες δοκιμές με το sqlmap στραφήκαμε σε ελληνικούς προορισμούς. Σύντομα, συναντήσαμε ένα site που δεν είχε κανέναν ενδοιασμό να μας αναφέρει τις βάσεις του...

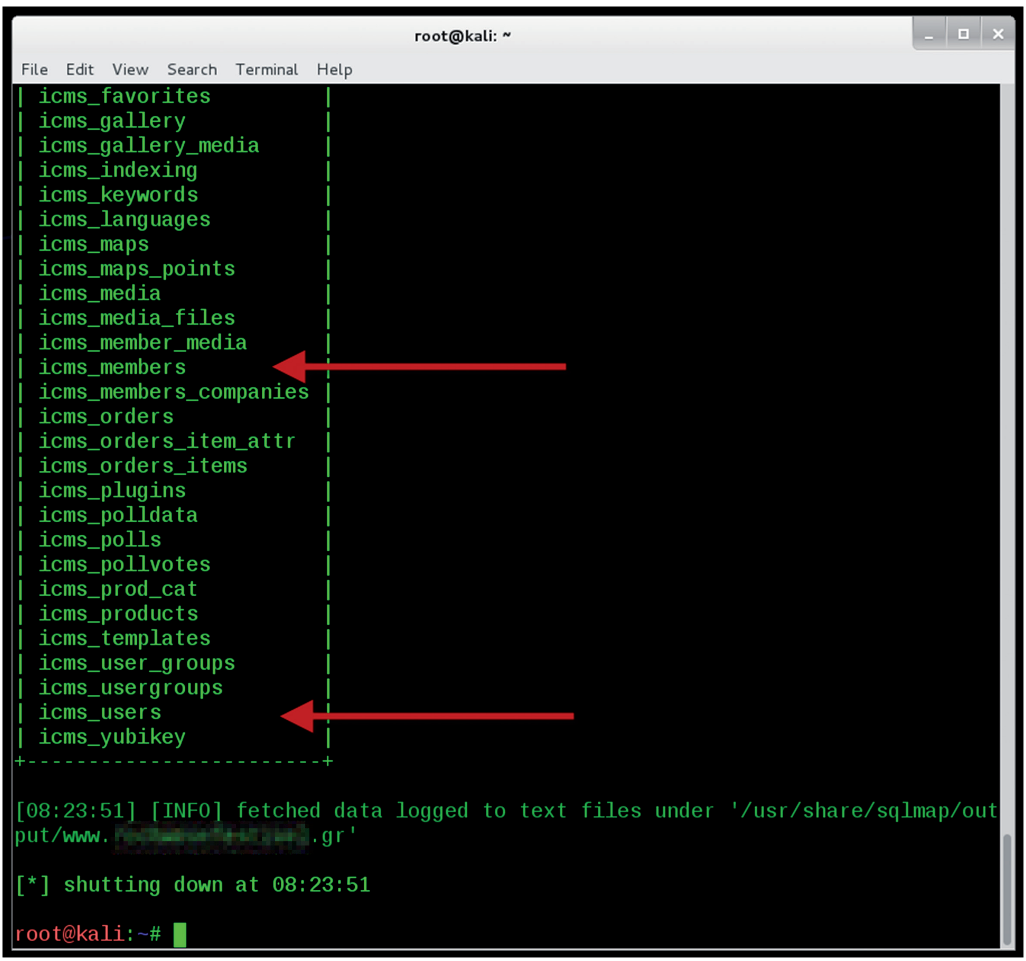
δυσεύρετο. Για παράδειγμα, ας υποθέσουμε ότι θέλουμε να επιτεθούμε σε μία φόρμα ενός site, που μεταφέρει δεδομένα με τη μέθοδο GET. Γι' αυτό το σκοπό, αρκεί να συμπληρώσουμε τη φόρμα με τυχαία δεδομένα και να την υποβάλουμε. Στη συνέχεια, μπορούμε να αντιγράψουμε το URL που σχηματίστηκε κατά την υποβολή της φόρμας και να τρέξουμε το sqlmap κάπως έτσι:

```
sqlmap --url 'www.some-poor-site.com/?var_name1=xxx&var_name2=yyy...'
```

Εναλλακτικά, αν το site δεχόταν δεδομένα μέσω της μεθόδου POST, θα έπρεπε να μάθουμε τα ονόματα των μεταβλητών και να εκτελέσουμε το sqlmap κάπως έτσι:

```
sqlmap --url 'www.some-poor-site.com' --data='var_name1=xxx,var_
name2=yyy...'
```

Όπως αναφέραμε προηγουμένως, το sqlmap ενσωματώνει ένα πλούσιο σύνολο λειτουργιών. Δεν σας κρύβουμε ότι όταν αρχίσαμε να παίζουμε μαζί του, ένα από τα



Στο ίδιο site καταφέραμε να διαβάσουμε και τους πίνακες των βάσεων. Στην εικόνα έχουμε σημειώσει δύο από τους πιο ενδιαφέροντες: icsms_memebrs και icsms_users.

πρώτα πράγματα που κάναμε ήταν μερικές δοκιμές σε πραγματικά site. Φυσικά δεν προξενήσαμε την παραμικρή ζημιά ούτε επιμείναμε σε κάποια επίθεση, ώστε να δημιουργήσουμε προβλήματα στη λειτουργία των στόχων. Δείτε μερικές από τις δυνατότητες που δοκιμάσαμε:

```
sqlmap --url 'www.target.com/?param=xxx' --dbs
```

Η παράμετρος --dbs ζητά από το sqlmap να ανιχνεύσει όλες τις βάσεις δεδομένων που διαχειρίζεται η εφαρμογή-στόχος.

```
sqlmap --url 'www.target.com/?param=xxx' -D db_name --tables
```

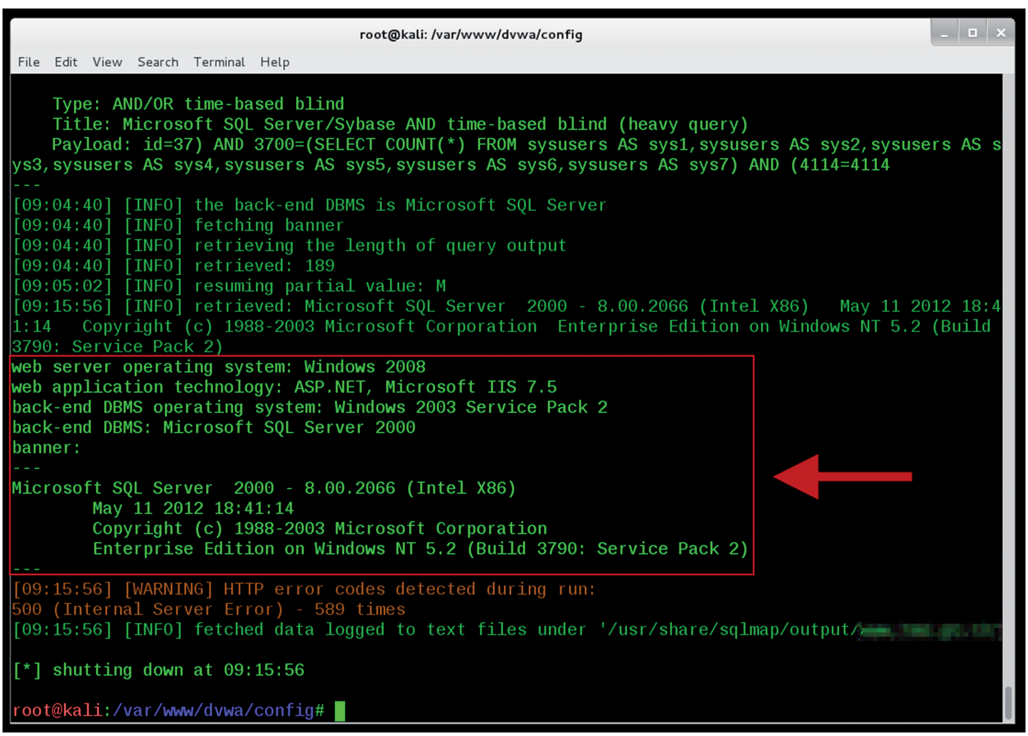
Δηλώνοντας μια συγκεκριμένη βάση με την παράμετρο -D, ζητήσαμε τα ονόματα όλων των πινάκων (--tables).

```
sqlmap --url 'www.target.com/?param=xxx' -D db_name -T table_name
--columns
```

Προσδιορίζοντας μια συγκεκριμένη βάση κι έναν συγκεκριμένο πίνακα (παράμετροι -D και -T), ζητήσαμε τα ονόματα των πεδίων (--columns).

```
sqlmap --url 'www.target.com/?param=xxx' --passwords --threads 4
```

Αν και ο "χειροκίνητος" εντοπισμός πινάκων και δεδομένων είναι πιο αποτελεσματικός, το sqlmap μπορεί να ψάξει για συγκεκριμένα δεδομένα από μόνο του. Για παράδειγμα, μπορεί να αναζητήσει τα ονόματα των χρηστών (--users) ή ακόμα και τα


```
root@kali: /var/www/dvwa/config
File  Edit  View  Search  Terminal  Help

    Type: AND/OR time-based blind
    Title: Microsoft SQL Server/Sybase AND time-based blind (heavy query)
    Payload: id=37) AND 3700=(SELECT COUNT(*) FROM sysusers AS sys1,sysusers AS sys2,sysusers AS s
ys3,sysusers AS sys4,sysusers AS sys5,sysusers AS sys6,sysusers AS sys7) AND (4114=4114
---
[09:04:40] [INFO] the back-end DBMS is Microsoft SQL Server
[09:04:40] [INFO] fetching banner
[09:04:40] [INFO] retrieving the length of query output
[09:04:40] [INFO] retrieved: 189
[09:05:02] [INFO] resuming partial value: M
[09:15:56] [INFO] retrieved: Microsoft SQL Server  2000 - 8.00.2066 (Intel X86)   May 11 2012 18:4
1:14   Copyright (c) 1988-2003 Microsoft Corporation  Enterprise Edition on Windows NT 5.2 (Build
3790: Service Pack 2)
web server operating system: Windows 2008
web application technology: ASP.NET, Microsoft IIS 7.5
back-end DBMS operating system: Windows 2003 Service Pack 2
back-end DBMS: Microsoft SQL Server 2000
banner:
---
Microsoft SQL Server  2000 - 8.00.2066 (Intel X86)
        May 11 2012 18:41:14
        Copyright (c) 1988-2003 Microsoft Corporation
        Enterprise Edition on Windows NT 5.2 (Build 3790: Service Pack 2)
---
[09:15:56] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 589 times
[09:15:56] [INFO] fetched data logged to text files under '/usr/share/sqlmap/output/[redacted]

[*] shutting down at 09:15:56

root@kali:/var/www/dvwa/config#
```

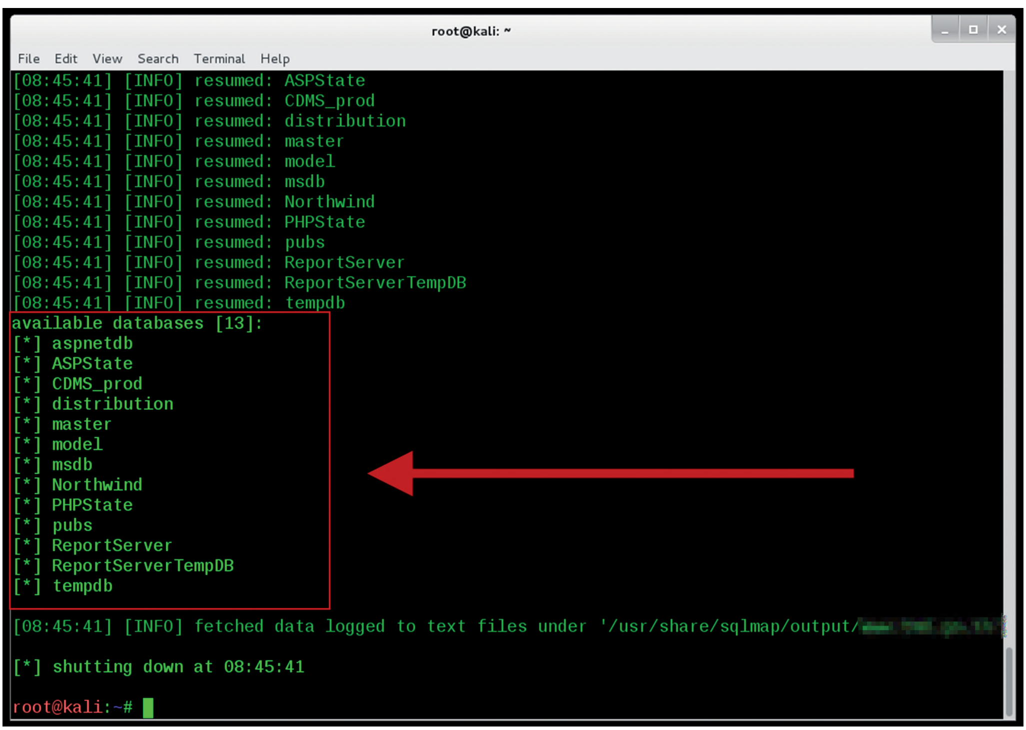
Μετά από τις δοκιμές σε ντόπια sites, στραφήκαμε στη μακρινή Ασία. Το Ταϊλανδέζικο site που αποφασίσαμε να σκαλίσουμε στηριζόταν σε server με τα Windows 2008, τον Microsoft IIS και τον Microsoft SQL server. Ναι, το sqlmap προσφέρει και τέτοιες πληροφορίες.

password (--passwords). Με την παράμετρο --threads δηλώνουμε πόσα threads θέλουμε να επιστρατεύσει το πρόγραμμα, προκειμένου να επισπεύσει την εργασία του.

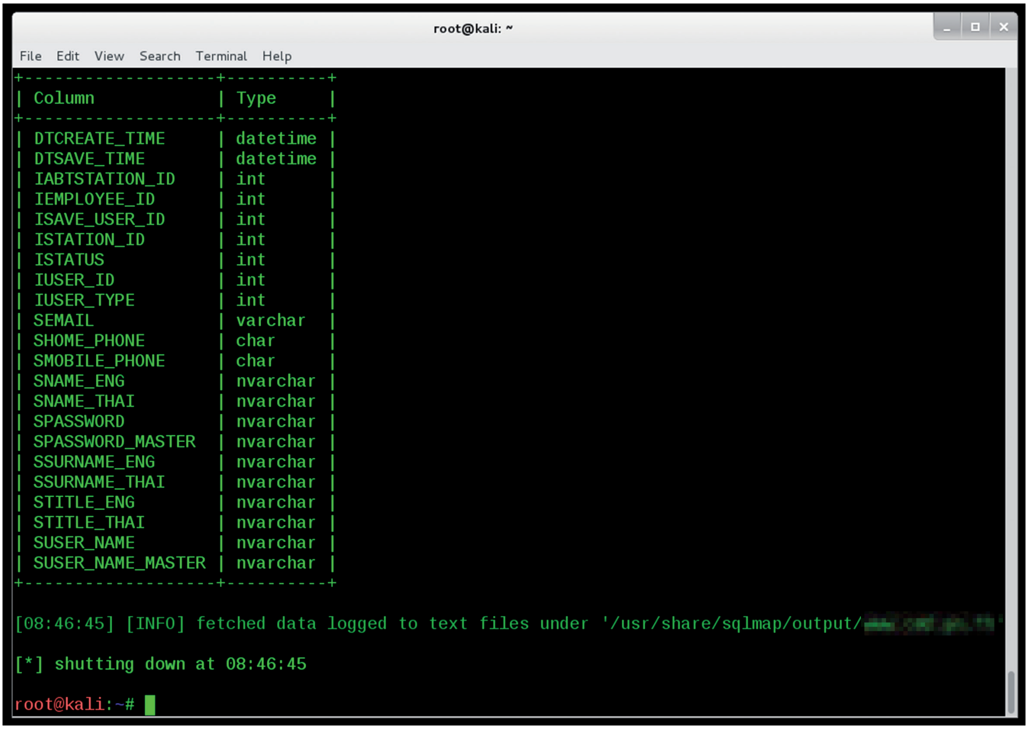
Μια πλήρης παράθεση των λειτουργιών θα ήταν κουραστική και μάλλον περιττή. Μπορείτε να μελετήσετε τις παραμέτρους που δέχεται το sqlmap από την τεκμηρίωσή του (https://github.com/sqlmapproject/sqlmap/wiki/Usage). Επιγραμματικά, θα αναφέρουμε μόνο ότι το sqlmap μπορεί ακόμα και να δημιουργήσει αρχεία στο λειτουργικό σύστημα του απομακρυσμένου server, να εκτελέσει εντολές και προγράμματα στο κέλυφος ή και να φυτέψει εκεί ένα shell. Φυσικά, όπως συμβαίνει με όλα τα εργαλεία αυτοματοποιημένων επιθέσεων, η επιτυχία αυτών των λειτουργιών δεν είναι εγγυημένη. Ωστόσο η ύπαρξή τους και μόνο καθιστά το sqlmap πολύτιμο εργαλείο για τον έλεγχο μιας εφαρμογής — της δικής μας εφαρμογής, για να μην ξεχνιόμαστε. Κι επειδή ξέρουμε ότι τα παραδείγματα που παρουσιάσαμε νωρίτερα ήταν κάπως ξερά και βαρετά, δείτε τις εικόνες στους άρθρου για να πάρετε μια γεύση από τα δεδομένα που καταφέραμε να ψαρέψουμε. Το sqlmap χρησιμοποιείται πανεύκολα ακόμα κι από κάποιον αρχάριο, οφείλουμε ωστόσο να υπογραμμίσουμε ότι δεν είναι παιχνιδάκι.

Εικονικός πόλεμος – αληθινή γνώση

Φτιάξατε μια εφαρμογή και πιστεύετε ότι λάβατε όλα τα απαραίτητα μέτρα, ώστε να μην είναι ευπαθής κατά SQL injection; Πολύ ωραία, όμως δεν χάνετε και τίποτε να δοκιμάσετε τις αντοχές της στην πράξη. Σηκώστε την εφαρμογή σε κάποιον



Ούτε κι αυτό το site ξεγλίστρησε από τις ανακριτές μεθόδους του sqlmap! Εδώ οι βάσεις ήταν περισσότερες και δυσκολευτήκαμε να επιλέξουμε στόχο...



```
root@kali: ~
File  Edit  View  Search  Terminal  Help
+-------------------+----------+
| Column            | Type     |
+-------------------+----------+
| DTCREATE_TIME     | datetime |
| DTSAVE_TIME       | datetime |
| IABTSTATION_ID    | int      |
| IEMPLOYEE_ID      | int      |
| ISAVE_USER_ID     | int      |
| ISTATION_ID       | int      |
| ISTATUS           | int      |
| IUSER_ID          | int      |
| IUSER_TYPE        | int      |
| SEMAIL            | varchar  |
| SHOME_PHONE       | char     |
| SMOBILE_PHONE     | char     |
| SNAME_ENG         | nvarchar |
| SNAME_THAI        | nvarchar |
| SPASSWORD         | nvarchar |
| SPASSWORD_MASTER  | nvarchar |
| SSURNAME_ENG      | nvarchar |
| SSURNAME_THAI     | nvarchar |
| STITLE_ENG        | nvarchar |
| STITLE_THAI       | nvarchar |
| SUSER_NAME        | nvarchar |
| SUSER_NAME_MASTER | nvarchar |
+-------------------+----------+

[08:46:45] [INFO] fetched data logged to text files under '/usr/share/sqlmap/output/[redacted]'

[*] shutting down at 08:46:45

root@kali:~#
```

Η ανίχνευση των βάσεων διήρκεσε αρκετή ώρα. Έτσι, αφού επιλέξαμε μια βάση για να προχωρήσουμε στη διερεύνησή μας, αποφασίσαμε να κάνουμε μια δοκιμή στα τυφλά. Αν και δεν είχαμε ιδέα για το αν υπάρχει πίνακας users, ζητήσαμε από το sqlmap να μας δώσει τις στήλες του. Τελικά, όπως βλέπετε στην εικόνα, σταθήκαμε τυχεροί. Η βάση που είχαμε επιλέξει διέθετε όντως έναν πίνακα με το όνομα users κι εκεί μέσα βρήκαμε αρκετά ενδιαφέροντα πεδία.

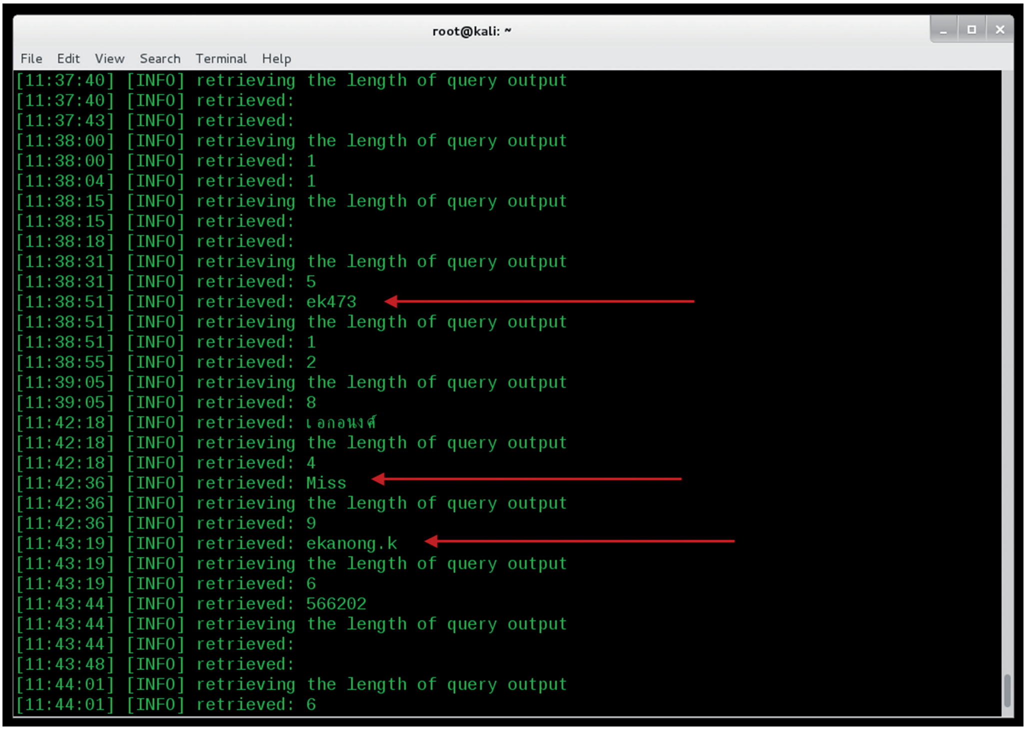
απομακρυσμένο server ή και τοπικά, εγκαταστήστε το sqlmap κι εξαπολύστε μια αδυσώπητη επίθεση. Στο τέλος των δοκιμών, θα έχετε σχηματίσει μια πιο ρεαλιστική άποψη για την αντοχή της εφαρμογής σας. Αν βέβαια το sqlmap καταφέρει να αλώσει την πόλη, τότε θα πρέπει να ανασκουμπωθείτε, να τσεκάρετε τον κώδικα — κι αν δεν βρείτε κάποια προφανή αδυναμία να το ρίξετε στη μελέτη. Το sqlmap ενημερώνει πάντα το χρήστη για το είδος των επιθέσεων που πραγματοποιεί, καθώς και για την επιτυχία τους. Εξάλλου, μπορείτε πάντα να το κάνετε πιο ομιλητικό, αυξάνοντας το verbosity level (παράμετρος –v). Αν εντοπίσετε κάποια αδυναμία στον κώδικά σας, μπορείτε να ξεκινήσετε τη μελέτη από την τεχνική που εφάρμοσε το sqlmap με επιτυχία.

Με τον τρόπο εργασίας που περιγράψαμε, είναι σίγουρο ότι το επίπεδο των γνώσεών σας θα βελτιώνεται διαρκώς. Επιπρόσθετα, εξασφαλίζοντας ότι η εφαρμογή σας μπορεί να σταθεί απέναντι στο sqlmap, θα μπορείτε να κοιμηθείτε (λίγο πιο) ήσυχοι. Βέβαια, υπάρχει και το σενάριο κατά το οποίο η εφαρμογή σας αντιστέκεται σε όλες τις επιθέσεις, από την πρώτη της έκδοση. Αυτό είναι σίγουρα ευχάριστο αλλά, ως γνωστόν, υπάρχουν και τα zero day attacks... Ποτέ δεν μπορείτε να είσαστε 100% σίγουροι για τη μελλοντική ασφάλεια της εφαρμογής σας. Μην το ψάχνετε, λοιπόν: Η διαρκής επαγρύπνηση (και μελέτη) αποτελεί μονόδρομο!

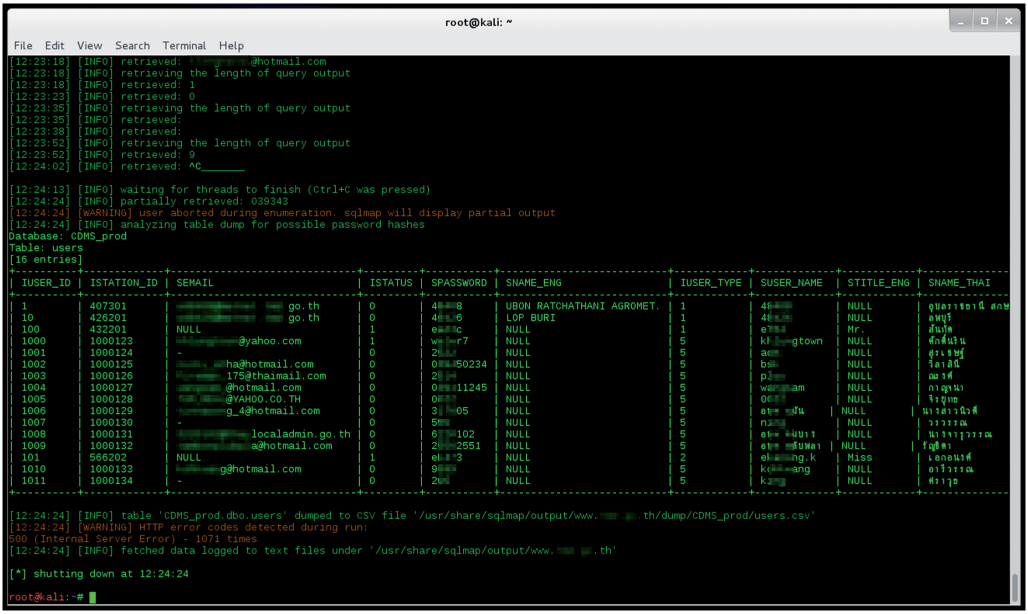
Εικονικό πεδίο μάχης

Όσο εξοικειώνεται κανείς με ένα γνωστικό αντικείμενο, τόσο πιο εύκολα μπορεί να παρακολουθήσει τις σχετικές εξελίξεις. Επιπρόσθετα, καθώς εμβαθύνει είναι ευκολότερο να διακρίνει τις καλές κι αξιόπιστες πηγές, από εκείνες που αναπαράγουν ανακρίβειες. Το ίδιο ισχύει και στη μελέτη των επιθέσεων SQL injection. Τι σημαίνουν όμως όλα αυτά για τους αρχάριους; Μήπως είναι καταδικασμένοι να ψάχνουν από 'δώ κι από 'κεί για συμβουλές, παραδείγματα και κατευθύνσεις για τη μελέτη τους; Η αλήθεια είναι πως, όχι, δεν χρειάζεται. Δίνουμε ένα παράδειγμα που στηρίζει τον ισχυρισμό μας.

Μια ομάδα προγραμματιστών που ασχολείται με την ασφάλεια λογισμικού έχει αναπτύξει την εφαρμογή Damn Vulnerable Web Application ή απλά DVWA (www.dvwa.co.uk). Πρόκειται για μια εφαρμογή που στηρίζεται στην PHP και στη MySQL και η οποία είναι ευάλωτη σε ένα μεγάλο σύνολο επιθέσεων (SQL injection, cross site scripting, cross site request forgery κ.ά.). Όπως αντιλαμβάνεστε, οι ευπάθειες δεν προέκυψαν τυχαία στον κώδικα της εφαρμογής, αλλά τοποθετήθηκαν εσκεμμένα. Το DVWA δημιουργήθηκε με σκοπό να αποτελέσει ένα εικονικό περιβάλλον εξάσκησης, για όσους θέλουν να μελετήσουν τα διάφορα είδη επιθέσεων και ν' αναπτύξουν τα κατάλληλα αντίμετρα. Ο κώδικας του DVWA προβλέπει δύο επίπεδα δυσκολίας για κάθε επίθεση, ενώ διαθέτει κι ένα τρίτο επίπεδο, στο οποίο οι επιθέσεις αποκρούονται αποτελεσματικά. Έτσι, οι μαθητευόμενοι έχουν στη διάθεσή τους παραδείγματα προς



Περιττό να πούμε ότι όταν είδαμε τις στήλες του πίνακα users, μας κυρίευσε ο σατανάς. Στην εικόνα βλέπετε στιγμιότυπο από τη δράση του sqlmap, καθώς διαβάζει τα περιεχόμενα του εν λόγω πίνακα.



Ο πίνακας users ήταν τεράστιος! Σύμφωνα με το sqlmap διέθετε κάτι παραπάνω από χίλιες εγγραφές. Φυσικά, εμείς δεν είχαμε κακή πρόθεση και το μόνο που μας ενδιέφερε ήταν να δοκιμάσουμε τις δυνατότητες του sqlmap. Ακριβώς γι' αυτό, διακόψαμε την εργασία του πολύ σύντομα και, τότε, εμφανίστηκε ένας πίνακας τα δεδομένα που είχε προλάβει να αλιεύσει.

αποφυγήν, καθώς και παραδείγματα προς μίμηση. Φυσικά, οι δημιουργοί του DVWA δεν διατείνονται ότι έγραψαν τον καλύτερο δυνατό κώδικα. Ακόμα και τα τμήματα που αποκρούουν τις επιθέσεις θα μπορούσαν να γίνουν πιο αποτελεσματικά ή και πιο αποδοτικά. Σε κάθε περίπτωση, το DVWA αποτελεί ένα ενδιαφέρον περιβάλλον για πειράματα και μελέτη.

Εγκατάσταση και ρύθμιση

Εμείς δοκιμάσαμε το DVWA στο Kali, αλλά μπορεί να στηθεί εξίσου άνετα σε οποιοδήποτε λειτουργικό. Τα μόνα προαπαιτούμενα είναι η παρουσία ενός web server, του interpreter της PHP και του MySQL server. Όλα αυτά υπάρχουν προεγκατεστημένα σε πολλές διανομές Linux, ενώ αν δεν υπάρχουν προστίθενται πανεύκολα και με μία κίνηση: Εγκαθιστώντας το πακέτο LAMP στο Linux ή το WAMP στα Windows. Από 'κεί και πέρα, αρκεί να ενεργοποιήσουμε τις σχετικές υπηρεσίες, να κατεβάσουμε τον κώδικα του DVWA, να τον αποσυμπιέσουμε στο Document Root του web server και να ακολουθήσουμε δύο απλά βήματα. Βλέπετε, το DVWA απαιτεί την επικοινωνία με τη MySQL, όπως και την ύπαρξη συγκεκριμένων βάσεων και πινάκων. Αυτό που χρειάζεται από μέρους μας είναι να ανοίξουμε το αρχείο ρυθμίσεων και να συμπληρώσουμε τα στοιχεία ενός χρήστη της MySQL. Το αρχείο ονομάζεται config.inc.php και βρίσκεται στον κατάλογο /var/www/dvwa/config[2]. Μετά απ' αυτό, μπορούμε να ανοίξουμε έναν web browser και να τον κατευθύνουμε στον τοπικό web server. Εκεί θα μας προϋπαντήσει το login prompt του DVWA. Το προκαθορισμένο όνομα χρήστη που δέχεται η εφαρμογή είναι το admin και το αντίστοιχο συνθηματικό είναι

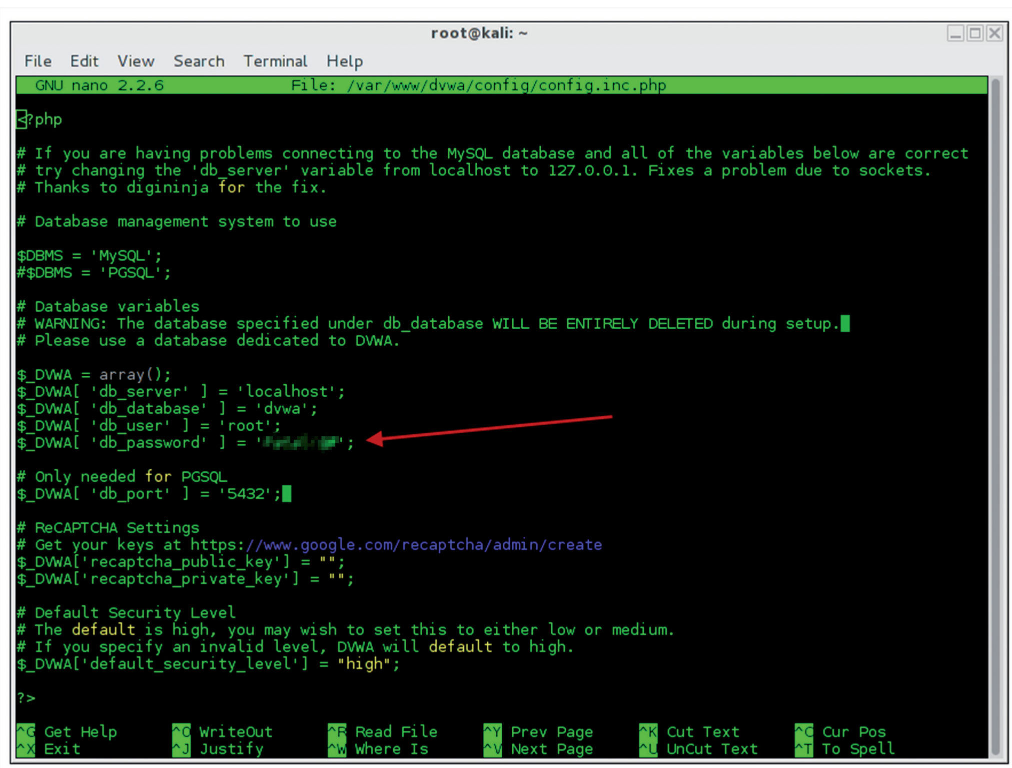
2. Όπως είναι φανερό, το πρώτο τμήμα της διαδρομής αποτελεί απλά το Document Root του web server.

το "password" (χωρίς τα εισαγωγικά). Αφού συνδεθούμε, αρκεί να μεταβούμε στην περιοχή setup (υπάρχει σύνδεσμος στο αριστερό μενού της εφαρμογής) και να πατήσουμε το κουμπί Create / Reset Database. Αυτό ήταν όλο. Στο εξής θα έχουμε στη διάθεσή μας έναν εικονικό αντίπαλο, με τρία επίπεδα δυσκολίας. Παρεμπιπτόντως, για να ρυθμίσουμε το πόσο ευπαθής θα είναι η εφαρμογή —ή το αν θα αποκρούει τις επιθέσεις μας— αρκεί να μεταβούμε στην περιοχή DVWA security. Εκεί θα βρούμε ένα drop-down menu με τα τρία επίπεδα ασφάλειας και το κουμπάκι υποβολής της επιλογής μας.

Πρώτη επαφή

Αφού εγκαταστήσουμε το DVWA, μπορούμε να ξεκινήσουμε τα πειράματα αμέσως. Για τη μελέτη των επιθέσεων SQL injection, η εφαρμογή διαθέτει δύο περιοχές: Την "SQL Injection" που προορίζεται για ένα σχετικά απλό και πιο κατανοητό είδος επιθέσεων, καθώς και την "SQL Injection (Blind)", που προσφέρεται για πιο προχωρημένες επιθέσεις κατά τις οποίες τα δεδομένα εξάγονται με την εις άτοπον απαγωγή. Όσοι ξεκινούν τώρα την ενασχόληση με το θέμα, θα ήταν προτιμότερο να στραφούν στην πρώτη περιοχή.

Μία ενδιαφέρουσα πρόκληση, για όσους θέλουν να κατανοήσουν πλήρως το πώς πραγματοποιούνται οι επιθέσεις SQL injection, θα ήταν να μελετήσουν τον κώδικα

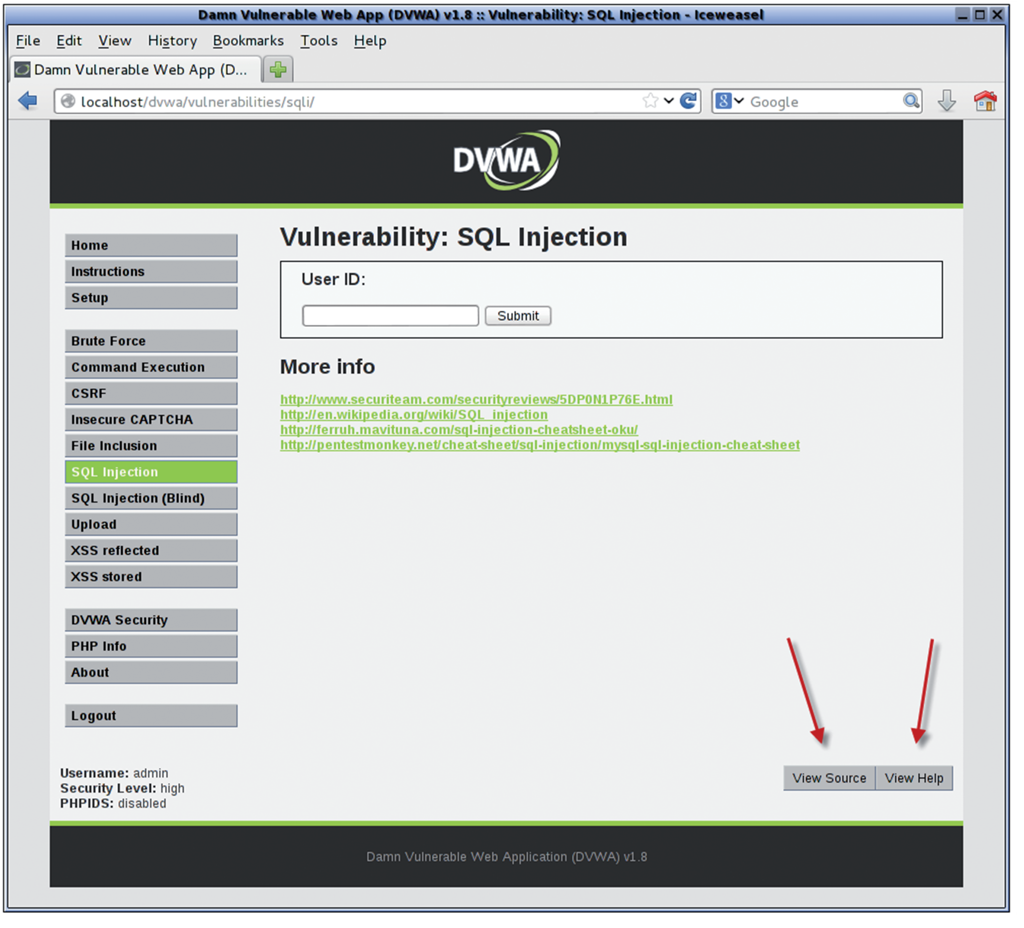


Το αρχείο ρυθμίσεων του DVWA είναι αρκετά μεγάλο και μοιάζει τρομακτικό. Ωστόσο, για τη βασική ρύθμιση της εφαρμογής αρκεί να συμπληρώσουμε τα στοιχεία ενός χρήστη της MySQL.

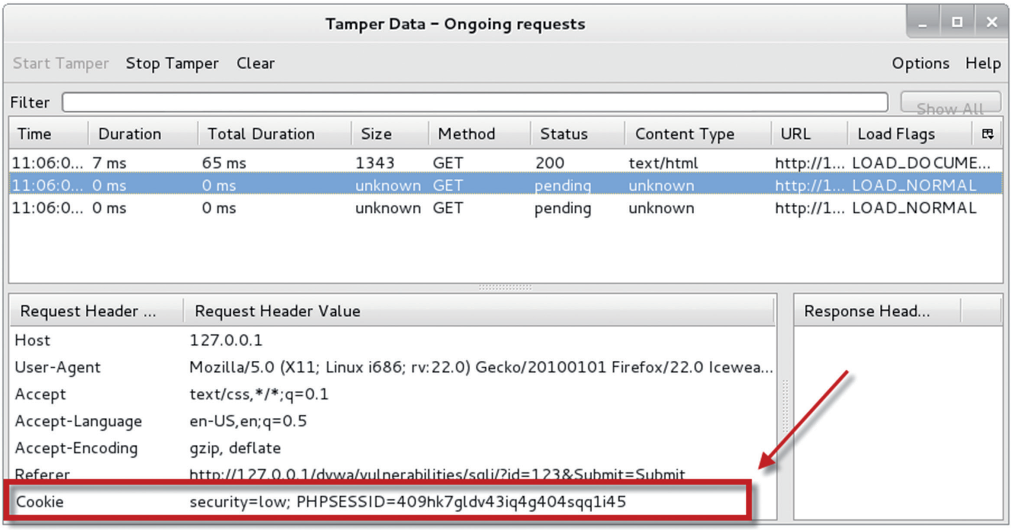
της σελίδας και να επιχειρήσουν την επίθεση χειροκίνητα. Εξάλλου, η ίδια η εφαρμογή προσφέρει δύο βολικά κουμπάκια, για την προβολή του πηγαίου κώδικα και για την εμφάνιση ενός σύντομου κειμένου βοήθειας. Εμείς θα αφήσουμε αυτό παιχνίδι σε εσάς και σας προτείνουμε να μην το εγκαταλείψετε, όσο κι αν δυσκολευτείτε στην αρχή. Στη συνέχεια θα εξετάσουμε το πώς μπορούμε να αξιοποιήσουμε το sqlmap, για να επιτεθούμε στο DVWA.

Αυτοματοποιημένη επίθεση

Όπως είπαμε προηγουμένως, για να παίξουμε με το DVWA πρέπει πρώτα να κάνουμε login. Αναρωτιέστε πώς θα πετύχουμε κάτι τέτοιο από το sqlmap; Όπως γνωρίζετε (ειδικά όσοι παρακολουθείτε τη σειρά άρθρων για το LAMP stack), μετά από μια επιτυχημένη σύνδεση η εκάστοτε εφαρμογή αποστέλλει στον web browser του



Το Damn Vulnerable Web Application διαθέτει δύο σελίδες για την πραγματοποίηση επιθέσεων SQL Injection. Στην εικόνα φαίνεται η σελίδα για τις "απλές" επιθέσεις (όχι τις Blind). Όπως βλέπετε, ο εκπαιδευτικός χαρακτήρας του DVWA δεν κρύβεται. Στην άκρη της σελίδας περιλαμβάνει κουμπάκια για την προβολή του πηγαίου κώδικα, καθώς και για την εμφάνιση ενός κειμένου βοήθειας.



Το πρόσθετο του Firefox ονόματι Tamper Data, εν δράσει! Στην εικόνα βλέπουμε το cookie που στέλνει το Damn Vulnerable Web Application στον web browser του χρήστη, μετά από μια πετυχημένη σύνδεση. Το περιεχόμενο του cookie το δώσαμε στο sqlmap για να εξαπολύσει μα επίθεση, πείθοντας το DVWA ότι έχει συνδεθεί κανονικά.

χρήστη ένα cookie. Με αυτόν τον τρόπο μεταφέρεται στο χρήστη ένα ειδικό κλειδί, το οποίο αποδεικνύει ότι έχουμε συνδεθεί. Ειδικά στην περίπτωση του DVWA, το εν λόγω cookie περιλαμβάνει και μια μεταβλητή η οποία δηλώνει το επιλεγμένο επίπεδο δυσκολίας. Επομένως, για να επιτεθούμε στην εφαρμογή από την γραμμή εντολών πρέπει να πάρουμε τα δεδομένα του cookie και να τα δώσουμε στο sqlmap. Αυτό επιτυγχάνεται πανεύκολα με διάφορα extensions που διατίθενται για κάθε browser. Στην περίπτωση του Firefox, ένα τέτοιο πρόσθετο είναι το λεγόμενο Tamper Data (https://addons.mozilla.org/en-US/firefox/addon/tamper-data). Αφού το εγκαταστήσουμε, αρκεί να το ενεργοποιήσουμε και να κάνουμε login στο DVWA. Στη συνέχεια, μπορούμε να πάμε στην περιοχή DVWA security, να επιλέξουμε ένα επίπεδο δυσκολίας και να πατήσουμε το κουμπί submit. Από τη στιγμή που ξεκινά το Tamper Data, καταγράφει όλα τα αιτήματα που στέλνει ο web browser προς τον server. Έτσι, αρκεί να μεταβούμε στο παράθυρο του πρόσθετου και να κάνουμε κλικ στο πιο πρόσφατο αίτημα, ώστε να δούμε τα δεδομένα που περιλαμβάνει. Εκεί θα βρούμε και το string που περιλαμβάνει το cookie. Έχοντας στη διάθεσή μας αυτά τα δεδομένα, είμαστε σχεδόν έτοιμοι να εξαπολύσουμε το sqlmap! Το τελευταίο βήμα είναι να μεταβούμε στην περιοχή SQL Injection, να κάνουμε μια δοκιμαστική υποβολή της φόρμας και να αντιγράψουμε το URL που σχηματίζεται. Βλέπετε, αν και το DVWA μεταφέρει ορισμένες παραμέτρους του μέσω του cookie, ο κώδικάς του περιμένει κι ορισμένες μεταβλητές μέσω της μεθόδου GET. Έχοντας και το σχετικό URL, μπορούμε να σαρώσουμε τη χαλαρή άμυνα της εφαρμογής κάπως έτσι:

```
sqlmap --cookie='PHPSESSID=lpb5g4uss9kp70p8jccjeks621; security=low'
       --url 'http://localhost:80/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit'
```

Προφανώς, αυτό που δώσαμε στην παράμετρο --cookie δεν είναι άλλο από το περιεχόμενο του cookie, που μας είχε στείλει η εφαρμογή DVWA. Αν πραγματοποιήσετε κι εσείς αυτή τη δοκιμή, θα διαπιστώσετε πόσο εξυπηρετικό είναι το sqlmap. Παρεμπιπτόντως, αν θέλετε να δοκιμάσετε τις δυνάμεις του στο επόμενο επίπεδο δυσκολίας του DVWA, αρκεί να τροποποιήσετε κατάλληλα τη σχετική μεταβλητή εντός του cookie. Αρκεί να δώσετε κάτι τέτοιο:

```
sqlmap --cookie='PHPSESSID=lpb5g4uss9kp70p8jccjeks621; security=medium'
       --url 'http://localhost:80/dvwa/vulnerabilities/
sqli/?id=1&Submit=Submit'
```

Για να αντιληφθείτε την ευκολία που προσφέρει το sqlmap στη διαδικασία των δοκιμών, σας προτείνουμε να επιτεθείτε στο DVWA χειροκίνητα. Να είσαστε σίγουροι ότι, ειδικά στο μεσαίο επίπεδο, θα αντιμετωπίσετε μεγάλη δυσκολία. Κι αυτό, για μια εφαρμογή που γνωρίζετε πολύ καλά ότι είναι ευπαθής.

Αντίμετρα

Εν ολίγοις, το μυστικό βρίσκεται στο σωστό και προσεκτικό input sanitization σε *κάθε* περιοχή του κώδικα που λαμβάνει δεδομένα από το χρήστη. Εδώ χρειάζεται λίγη προσοχή. Βλέπετε, μερικοί αρχάριοι προγραμματιστές καταντούν να ελέγχουμν αν τα δεδομένα περιλαμβάνουν απαγορευμένους χαρακτήρες ή όχι. Αυτή η τακτική, όμως, είναι αφελής και επικίνδυνη. Είναι βλέπετε αδύνατο να προβλέπουμε όλες τις τεχνικές —τις υπάρχουσες αλλά κι αυτές που θα ανακαλυφθούν— για την παράνομη εισαγωγή κώδικα SQL. Επομένως, αντί να ελέγχουμε για συγκεκριμένα χαρακτηριστικά, πρέπει να *επιβάλλουμε* τη μορφή που θεωρούμε αποδεκτή! Για το σκοπό αυτό η PHP διαθέτει πολλές εξειδικευμένες συναρτήσεις, όπως η addslashes(), η htmlentities() και η mysql_real_escape_string(). Μάλιστα η τρίτη από αυτές είναι μάλλον καταλληλότερη για το θέμα που εξετάζουμε, αφού φροντίζει να "απενεργοποιήσει" τους χαρακτήρες που έχουν ειδική ερμηνεία στη γλώσσα SQL. Μια άλλη καλή τακτική είναι αυτή της μετατροπής (cast) μιας μεταβλητής σε συγκεκριμένο τύπο. Αυτή η μέθοδος βρίσκει εφαρμογή σε μεταβλητές που οφείλουν να έχουν αποκλειστικά αριθμητικό περιεχόμενο. Για παράδειγμα, όταν η μεταβλητή $my_var αναμένεται να έχει αριθμητική τιμή κι οτιδήποτε άλλο θα αποτελούσε απόπειρα εξαπάτησης ή σφάλμα, μπορούμε να κάνουμε κάτι τέτοιο:

```
$sanitized_var = intval($my_var);
```

Όπως καταλαβαίνετε, για την αντιμετώπιση των επιθέσεων SQL δεν υπάρχουν καθολικές λύσεις, ούτε μαγικές συνταγές. Υπάρχουν μόνον ορισμένες κατευθυντήριες γραμμές. Από εκεί και πέρα, κάθε προγραμματιστής που φιλοδοξεί να δημιουργεί ασφαλείς εφαρμογές οφείλει να ενημερώνεται και να μελετά αδιάκοπα. Ευτυχώς που για 'μάς η μελέτη προγραμματιστικών ζητημάτων αποτελεί διασκέδαση ;)


COOKIES
ΦΑΡΜΑΚΕΙΟ
ΓΚΑΧΑ
ΓΚΟΥΧΑ
ΑΚΕΙΟ
BEAUTE
ΓΕΙΑ ΣΑΣ... ΑΥΤΑ ΕΔΩ ΘΑ ΗΘΕΛΑ...
ΣΥΝΤΑΓΗ
ΜΑΛΙΣΤΑ... ΑΝΤΙΜΥΞ ΤΩΝ 500, ΠΥΡΕΤΟΛ, ΒΗΧΟΣΤΟΠ ΣΙΡΟΠΙ...
ΙΩΣΟΥΛΑ, ΙΩΣΟΥΛΑ;
2 ΦΟΡΕΣ ΤΟ ΚΑΘΕΝΑ, ΓΙΑ ΜΙΑ ΒΔΟΜΑΔΑ ΚΑΙ ΘΑ'ΣΑΙ ΜΙΑ ΧΑΡΑ!
ΟΛΑ ΜΑΖΙ ΕΙΝΑΙ 27 ΕΥΡΩ ΚΑΙ 60 ΛΕΠΤΑ!...
ΠΛΙΚ ΠΛΙΚ
VITAMIN
TOTAL
ΑΦΟΥ ΕΙΝΑΙ ΜΟΝΟ ΓΙΑ ΜΙΑ ΒΔΟΜΑΔΑ...
...ΜΗΠΩΣ ΝΑ ΔΟΚΙΜΑΖΑ THN FREE TRIAL VERSION;